Aalto University
School of Science
Master's Programme in Mathematics and Operations Research

Katri Selonen

An Adaptive Recommender System for News Delivery

Master's Thesis
Espoo, July 31, 2017

Supervisor: Prof. Ahti Salo
Advisor: Teemu Kinnunen, D.Sc. (Tech.)

Aalto University
School of Science
Master's Programme in Mathematics and Operations Research

ABSTRACT OF
MASTER'S THESIS

Author	Katri Selonen		
Title	An Adaptive Recommender System for News Delivery		
Major	Systems and Operations Research	**Code**	SCI3055
Supervisor	Prof. Ahti Salo		
Advisor	Teemu Kinnunen, D.Sc. (Tech.)		
Date	July 31, 2017	**Pages**	vi + 65

Modern news websites contain plenty of constantly changing content. To better cater for different user types, the content shown needs to be personalised. This thesis presents the development of a recommender system for a large media company. The recommender system generates suggestions for news articles, and the personalised content is to replace the company's news site's current manually composed front page. The main goal is to increase user activity on the news site.

One of the challenges in creating a news recommender system is the vast amount of user and article data. Furthermore, the relevance of articles usually changes over time. As a result, there is a need for consideration of the cold-start problem and fast adaptability. Consequently, the chosen approach is memory-based. In this thesis, the preferences of each user are modelled by the users' visit frequency and the sections they read articles from. The recommender system can therefore be classified as content-based, with some context-based additions. As an addition, *article importance* scores are formed with a novel approach and are used as a basis for calculating scores for articles. They reflect the editorial view of articles and aid in maintaining the general feel of the news site.

The resulting recommender system is fast, lightweight, and can provide suggestions even with very little user transaction data. However, the results of the chosen performance metrics are inconclusive: for all tested parameter variants, some of the metrics show an increase in user activity while others show the opposite. The proposed next steps are to either do more online testing with different parameter combinations or to implement new features.

Keywords	recommender system, online news, content-based, context-based, memory-based
Language	English

Aalto-yliopisto
Perustieteiden korkeakoulu
Matematiikan ja operaatiotutkimuksen maisteriohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä	Katri Selonen		
Työn nimi	Uutisten välityksen adaptiivinen suositusjärjestelmä		
Pääaine	Systeemi- ja operaatiotutkimus	**Koodi**	SCI3055
Valvoja	Prof. Ahti Salo		
Ohjaaja	TkT Teemu Kinnunen		
Päiväys	31. heinäkuuta 2017	**Sivumäärä**	vi + 65

Modernit uutissivustot sisältävät runsaasti jatkuvasti muuttuvaa sisältöä. Erilaisten käyttäjien tarpeisiin vastataan paremmin personoimalla näytetty sisältö. Diplomityössä kehitetään suositusjärjestelmä suurelle mediayhtiölle. Suositusjärjestelmä ehdottaa uutisartikkeleita käyttäjille. Nämä ehdotukset tulevat korvaamaan yhtiön uutissivuston nykyisen etusivun. Päätavoiteena on kasvattaa sivuston käyttöä.

Yksi uutisten suosittelujärjestelmän luomiseen liittyvistä haasteista on käyttäjä- ja artikkelidatan suuri määrä. Lisäksi artikkelien merkityksellisyys muuttuu yleensä ajan myötä. Tämän takia erityisesti kylmäkäynnistys-ongelmaan sekä nopeaan adaptiivisuuteen on kiinnitettävä huomiota. Järjestelmään on siksi valittu muistipohjainen lähestymistapa. Diplomityössä käyttäjien mieltymyksiä mallinnetaan heidän käyttöuseuden sekä luettujen artikkelien osioiden avulla. Suosittelujärjestelmää voidaan siten kutsua sisältöpohjaiseksi, kontekstipohjaisin lisäyksin. Käyttäjäprofiilin lisäksi suositusten muodostamista varten luodaan uudenlainen *artikkelin tärkeys* -arvo. Se heijastelee toimituksen näkemyksiä ja auttaa ylläpitämään sivuston tuntuman.

Luotu suositusjärjestelmä on nopea, kevyt ja tekee suositteluja jo hyvin pienellä määrällä dataa. Valituilla mittareilla saatujen tulosten perusteella ei kuitenkaan pystytä kiistatta sanomaan, että sivuston käyttö kasvaisi. Kaikilla kokeilluilla parametrivaihtoehdoilla mittareista osan mukaan käyttö kasvaa ja toisten mukaan vähenee. Seuraaviksi kehityskohteiksi ehdotetaan lisää käyttäjätestejä tai uusien ominaisuuksien kehittämistä parempien tulosten tavoittelemiseksi.

Asiasanat	suositusjärjestelmä, verkkouutiset, sisältöpohjainen, kontekstipohjainen, muistipohjainen
Kieli	Englanti

Acknowledgements

First, I want to thank my advisor Teemu Kinnunen. He provided more support than I even could have hoped for, both in trying to find a topic, as well as during the planning, implementation, and writing of this thesis. He also gave invaluable insight and feedback to help me improve my work.

Secondly, I would like to thank Futurice for the opportunity and the setting for working on my thesis and the compelling topic. I would like to give special thanks to my colleague Antti Vuorela for valuable discussions and technical support in the implementation phase.

Thirdly, I would like to thank Aalto University and the Systems Analysis Laboratory for providing interesting courses that taught me great tools to use in my future work and life. I would also like to thank my supervisor Ahti Salo for taking time to meticulously read my work and suggest improvements.

Last but not least, many thanks for Olli Niskanen, Victoria Eklund, Emmi Jokinen, and other friends and family for their immense support throughout my studies.

Espoo, July 31, 2017

Katri Selonen

Contents

Chapter 1

Introduction

There is an abundance of content available throughout the web. Finding the most interesting items, such as videos, clothes, or articles, can be a struggle. The amount of accessible content is ever increasing, and the relevance of items often changes over time. For example, clothes go out of fashion, and few people are interested in a car accident that happened a month ago.

Recommender systems help navigate the vast sea of items by finding content that corresponds to the users' preferences. Recommender systems provide individualised suggestions for items to users based on ratings estimated for those items (Adomavicius and Tuzhilin, 2005). Recommender systems have been proven to be very valuable (Ricci et al., 2015). Consequently, many websites — from e-commerce to news sites and everything in between — use or even depend on recommender systems, examples by Schafer et al. (1999), Linden et al. (2003), and Thurman and Schifferes (2012).

Recommender systems face many challenges because each use case is unique to some extent and the number of requirements increases as the systems get more sophisticated. Recommending news is particularly demanding, because of the diverse nature and rapidly changing relevance of news articles (Liu et al., 2010). Consequently, many of the items have not been seen by a particular user. This lack of data makes it harder to estimate the ratings

needed for providing suggestions. Moreover, user data is often hard to acquire without hampering the user experience, and explicitly collected data is rather sparse. On the other hand, implicit data is not complete either and comes with some uncertainty (Pazzani and Billsus, 2007). The lack of data of items or users in the beginning is called the cold-start problem (Ricci et al., 2015). It requires special attention in news recommender systems, because in the news domain it is important to provide timely suggestions (Li et al., 2011).

Different kind of recommender systems address these challenges. The most suitable type also depends on the characteristics of the data that is to be used. There is no recommender system suitable for all use cases or one that would take care of all the weaknesses (Burke, 2002). There are different classifications for recommender systems, but one of the most common ones is to classify them according to their approach to rating estimation (Adomavicius and Tuzhilin, 2005). The division is made between content-based and collaborative filtering, with a third group for hybrid recommender systems.

Content-based recommender systems make suggestions based on descriptions of the items and a profile of the user's interests (Pazzani and Billsus, 2007). For example, if the user has given a high rating to an Indian restaurant serving organic food, other Indian or organic restaurants will be recommended. Collaborative filtering, on the other hand, recommends items based on ratings given by similar users (Sarwar et al., 2001). For example, if the user were to give a high rating to an Indian restaurant, and others who liked that Indian restaurant gave a high rating to a nearby Nepalese restaurant, that Nepalese restaurant will be recommended to the user.

Different types of recommender systems face various challenges. Hybrid recommender systems combine different recommending techniques to mitigate the biggest challenges (Burke, 2007). However, there are also many common issues that affect almost all recommender systems. The most noteworthy of them is the difficulty of evaluating the goodness of the generated suggestions. This is because the performance of a recommender system is mostly based on users' opinions, which are hard to validate (Beliakov et al., 2011).

1.1 Approach and Scope

This thesis focuses on the development of a recommender system for a news website. Unlike many recommender systems that have been implemented as an additional box on the site, this recommender system is to replace the main content on the existing front page. The current front page is the same for all users, and is manually composed by a handful of people: the online news editors in charge, called DJs.

The recommender system is based on the requirements given by the news company and insight from the DJs. Deriving from those, the approach used is content-based with some context-based additions. Collaborative filtering is ruled out, because the freshness of articles is pivotal, and general lists of most popular articles are already provided to users elsewhere on the site.

The recommender system's ratings are based on an article importance model that is derived from the hand-composed front page. These article importance scores can be interpreted as implicit editorial ratings of the articles. The approach is quite novel and therefore it receives special focus in this thesis.

Implicit user data is collected of site visits and read articles for forming user profiles. No text-based methods like topic modelling are used. Instead, item data consists of manually set section and time stamp data. Because the content and users of the news website are constantly changing, the system needs to adapt fast to these changes. Therefore, a memory-based approach is used. Finally, the suggestions are selected and ordered by scores that are calculated from the article importance, user preferences, and other available data. They are presented to the user as a list of items that act as previews of articles.

1.2 Research Questions

The main goal of this thesis is to build a recommender system that increases user activity on the news site. The aim is to better cater for different types of users to increase revenue ultimately. However, there are many restrictions that limit the development choices. Therefore, an unconventional approach to forming the ratings is used. The main research question is, how valuable the results are in comparison with those of other recommender systems.

Another point of focus is the novel idea of the article importance scores that reflect the current feel of the site, instead of asking the editors or DJs to assign new values. This way, the system can be put online faster, and the news editors' job descriptions do not change.

1.3 Thesis Structure

Chapter 2 presents earlier research on different types of recommender systems and commonly used techniques, with focus on the strengths and weaknesses. Also, the common challenges that recommender systems face especially in the news domain are outlined. In Chapter 3, the chosen methods and the motivation for choices are presented. Chapter 4 describes the specifics of developing the recommender system in this use case, such as data and formulas used. Experiments run with the recommender system and the corresponding results are presented in Chapter 5. In Chapter 6, the results are discussed and the process of developing the recommender system is reviewed. Finally, Chapter 7 summarises the work and its implications.

Chapter 2

Background

This chapter presents recommender systems in more detail, considers different types of recommender systems, and addresses their advantages and disadvantages. Then, challenges common for all recommender systems are explored and additional remarks are given for matters which are particularly specific to recommending news.

2.1 Recommender Systems

Early recommendations include, among others, book recommendations from friends, employees' recommendations letters, and printed restaurant guides (Resnick and Varian, 1997). These are manual suggestions that are based on the opinions of a single person or very few people. Recommender systems are highly automated software tools for providing suggestions to aid users in finding relevant items (Ricci et al., 2015).

Recommender systems have existed since the mid-1990s to help with the growing amount of content (Adomavicius and Tuzhilin, 2005). Goldberg et al. (1992) introduced the term *collaborative filtering*, which described the system quite literally. Others adopted the term, but Resnick and Varian

(1997) wanted the term *recommender system* to be used instead, because not all systems incorporated collaborative components nor did only filtering. The web has provided access to continually more and more items, increasing the need for recommender systems, and therefore they have largely developed parallel with the web (Bobadilla et al., 2013).

Early recommender systems aggregated recommendations provided by other people and directed them to similar recipients (Resnick and Varian, 1997). Non-personalised suggestions, such as lists of most liked items, are easier to generate automatically, and systems that do that are quite common (Schafer et al., 1999). However, those are not addressed by the recommender system research. Resnick and Varian (1997) assert that when people's preferences differ, personalisation becomes more valuable.

Recommender systems make personalised suggestions for items that are predicted to be most suitable based on user preferences and other possible constraints. Predicting suitability is the main feature of recommender systems (Ricci et al., 2015), and it is usually done by estimating ratings of items the user has not yet seen (Adomavicius and Tuzhilin, 2005). The items with highest estimated ratings are then recommended to the user, for instance as a ranked list.

Recommender systems can be used in many fields and for various items such as music, books, documents, clothes, or TV programs, for example (Park et al., 2012). A single recommender system usually focuses on a certain type of item to be able to provide useful and effective suggestions. Moreover, the components and attributes of a recommender system are customised to fit the exact use case and data available (Ricci et al., 2015). Therefore, implementing a recommender system always requires special effort. Despite this effort, recommender systems are very popular (Bobadilla et al., 2013). There are many well-known examples of important recommender systems, such as the recommender systems of the Amazon.com, eBay, Levis, Netflix, and Tripadvisor websites (Ricci et al., 2015; Schafer et al., 1999).

Recently, the use of recommender systems has dramatically increased as they have proven to help with the information overload (Ricci et al., 2015). Additionally, new types of data, such as social or contextual information of the user, has been integrated in recommender systems (Bobadilla et al., 2013). According to Pazzani and Billsus (2007), the modern recommender system is often an interactive web application.

Usually, item, user, and transaction data are needed to generate suggestions. The transactions refer to the relations between users and items. Additionally, feedback on how the user responds to the suggestions can be collected to improve the future recommendations. The relations are often presented as ratings, which can be implicit or explicit (Ricci et al., 2015). Explicit data, which is specifically given by users, for example in the form of star ratings, increases the cognitive load, and may need an incentive to get users to provide it (Resnick and Varian, 1997). Therefore, especially in a large data set, explicit data can be very sparse. Implicit data can be collected without extra effort from users, for instance by interpreting buying an item as a positive preference (Bobadilla et al., 2013). However, because implicit data depends on interpretations, it often contains some noise (Pazzani and Billsus, 2007). The noise can be, for example, from accidentally opened item details. Furthermore, the type of data available often is restricted by the application domain.

The collected data is the source of the most universal challenge recommender systems face: the cold-start problem (Schein et al., 2002). Cold-start refers to situations in which there is none of the needed data on a user or an item and therefore suggestions cannot be generated. There are three different kinds of cold-start problems: new community, new user, and new item (Bobadilla et al., 2013). New community cold-start is faced when the recommender system is first implemented and the required data has not yet been collected. A profile cannot be formed for a user that has no recorded transactions, which is referred to as the new user cold-start problem. Similarly, item cold-start is encountered especially in collaborative filtering for items nobody has rated.

Methods used in different kind of recommender systems can be divided into memory-based and model-based according to their approach on handling data. Memory-based methods act on all data, whereas model-based methods use the data to learn a model that is used for generating recommendations (Adomavicius and Tuzhilin, 2005). Memory-based systems use all the accumulated data for generating results in real-time so they respond to changes in data immediately, but can suffer from scalability issues. In turn, model-based systems scale better, because the computationally expensive parts can be done offline, but new data needs to separately handled (Anand and Mobasher, 2003).

Different methods used in recommender systems include classification methods, like k-Nearest-Neighbours and Bayesian classifiers, clustering, decision trees, neural networks, regression, and other heuristic methods (Adomavicius and Tuzhilin, 2005; Park et al., 2012). The final results are usually provided as a probability or a binary value. The values describe whether the user will like the item, or a numeric value for example describing the user's degree of interest (Pazzani and Billsus, 2007).

Regardless of the chosen method, most recommender systems are based on maximising the user's utility. The utility is thought to be maximised by providing suggestions for items that have the highest estimated ratings for that user. Amatriain and Pujol (2015) point out that a similarity measure is usually used for estimating those ratings. Depending on the approach, the similarity is measured between pairs of items or users. Bobadilla et al. (2013) list common similarity measures: Euclidean distance, mean squared differences, cosine similarity, adjusted cosine, Pearson correlation, and constrained correlation. The most popular one of them is cosine similarity (de Gemmis et al., 2015).

Recommender systems are often classified into different types. The most common classification consists of content-based and collaborative filtering (Park et al., 2012). However, not all recommender systems fit nicely into these two classes — there are other types, like knowledge-based systems,

as well as hybrid systems that combine different types. The application domain and the desired outcome have a major effect on selecting the type of recommender system to use (Ricci et al., 2015; Schafer et al., 1999). These different types are presented in more detail in the following sections.

Building a recommender systems involves many technical choices and considerations. Matters affecting these choices include, for example, how homogeneous the users are, how to raise the users' trust in recommendations, or what is the cost of false positives (Resnick and Varian, 1997). It is necessary to look at the objective and desired quality of results, and how quality is defined in that use case (Bobadilla et al., 2013). Additionally, in many cases, the data needs to be preprocessed, using methods like structuring or dimensionality reduction (Amatriain and Pujol, 2015).

As recommender systems and computation power have developed, requirements for the systems have increased. Adaptability and taking the temporal nature of data into account have become increasingly common in recommender systems (Burke, 2007). Some other new concerns include data privacy (European Comission, 2016), and security against attacks, such as those that flood the system with large amounts of faulty data (Bobadilla et al., 2013). Moreover, there can be other case-specific issues related to building a recommender system, or to presenting the results.

2.2 Content-Based Recommender Systems

Content-based recommender systems suggest items based on item descriptions and the user's past choices (Pazzani and Billsus, 2007). The items most similar to the ones the user has previously liked are recommended (Ricci et al., 2015). The recommendation process usually consists of feature extraction or other content analysis, learning the user profiles, and filtering items based on item similarity calculated from the features associated with the items (Bobadilla et al., 2013).

In content-based recommending, a user profile is built based on the descriptions of items that interest the user and history of the user's transactions (Pazzani and Billsus, 2007). The model of the user's preferences is then used to predict future preferences (de Gemmis et al., 2015). However, user tastes can change over time, which creates an extra challenge.

Item data used for content-based filtering usually consist of item features that can be either from structured data or extracted from unstructured data, such as free text fields (de Gemmis et al., 2015). Pazzani and Billsus (2007) point out that user profiles are easy to learn from structured data, because the number of attributes is limited and they have a known set of values. More often than not, the structured data available is not sufficient so that feature extraction is needed (Burke and Ramezani, 2011). Unfortunately, analysing natural language suffers from ambiguity, caused for example by synonyms or change of meaning depending on context (Pazzani and Billsus, 2007).

Feature extraction is usually done with quite simple models, like keyword matching or the Vector Space Model (de Gemmis et al., 2015). In the Vector Space Model every document is represented by a vector of term weights. Term weights are often calculated using term frequency / inverse document frequency (TF-IDF) measure (Adomavicius and Tuzhilin, 2005). In TF-IDF, terms that are frequent in a document but rare in other documents are considered relevant to the topic of that document (de Gemmis et al., 2015).

The quality of the item features data will determine the quality of suggestions provided by a content-based recommender system, Burke and Ramezani (2011) remark. This is one of the main shortcomings of content-based filtering, and in some domains it is difficult to generate attributes to sufficiently distinguish items (Pazzani and Billsus, 2007). Another disadvantage of content-based systems is propensity for overspecialisation (Park et al., 2012). The systems are incapable of recommending anything different from what the user has seen before. This is reflected also to the new user cold-start problem: when there is very little data on the user, the system cannot provide accurate recommendations.

One advantage of content-based recommender systems is their user independence. User independence means that only the current user's data is needed for calculations instead of all users' data (de Gemmis et al., 2015). Similarly, because only the features of items are required, the systems can recommend new items that have not been rated by anyone. Basing recommendations on item features also makes it easier to provide explanations behind suggestions to users.

Burke and Ramezani (2011) note that content-based filtering is most suitable for applications in which items are related to quality and taste. Such items are, for instance, news, books, movies, restaurants, and job search recruiting. On the other hand, items that are seldom rated are not well-suited for content-based recommending because there is not enough transaction history to build a user profile comprehensive enough. According to Bobadilla et al. (2013), it is quite rare to have a pure content-based recommender system. Nonetheless, they are relatively easy to combine with other types, especially collaborative filtering (Pazzani and Billsus, 2007).

2.3 Collaborative Filtering

In collaborative filtering, item suggestions are based on what other users with similar tastes have liked (Adomavicius and Tuzhilin, 2005). It requires no additional content information on the items or users, but instead compares patterns in transaction histories (Ricci et al., 2015).

According to Koren and Bell (2015), the primary approaches for relating items and users in collaborative filtering are neighbourhood methods and latent factor models. Neighbourhood methods are more common of the two, with k-Nearest-Neighbours (kNN) being the most popular algorithm for collaborative filtering (Bobadilla et al., 2013). Neighbourhood algorithms are memory-based, whereas many other approaches, like latent factor models, Bayesian networks, and clustering are model-based (Sarwar et al., 2001).

Collaborative filtering also requires the use of a similarity measure. The system can use user-to-user or item-to-item relations to determine needed similarities (Koren and Bell, 2015). The former compares users' histories to find most similar users, and then suggests the items those users have liked. Item-based algorithms calculate item similarities using a user-item matrix to find which items are usually liked together (Sarwar et al., 2001).

Algorithms used for collaborative filtering are generally computationally expensive (Linden et al., 2003). However, Sarwar et al. (2001) point out that, if the item set is relatively static, item-to-item relations can be calculated offline to reduce online computation. An additional approach for the time intensiveness is to reduce the size of data with for example dimensionality reduction, but it decreases the quality of recommendations and is not suitable for adaptive data (Bobadilla et al., 2013).

In addition to scalability and performance issues caused by computation expense, collaborative filtering faces challenges with data sparsity and cold-start (Park et al., 2012). High sparsity means usually that there are few neighbours to compare to, which reduces recommendation accuracy. Moreover, if there are no ratings for an item, the system can not recommend that item (Bobadilla et al., 2013).

Collaborative filtering does not suffer from the need to distinguish items nor from overspecialisation, which are common in content-based filtering. Linden et al. (2003) praise collaborative filtering for its ability to help users find new and even surprising items. However, the desirability of that feature depends on the domain and purpose of the recommender system.

Collaborative filtering is suitable for similar application domains as content-based filtering (Burke and Ramezani, 2011), but it has different requirements for data. It is the most popular type of recommender system and has been widely used especially in e-commerce (Schafer et al., 1999). Koren and Bell (2015) adduce that as a result of the Netflix Prize competition, progress in the field of collaborative filtering has greatly advanced recently.

2.4 Other Recommender System Types

Ricci et al. (2015) list four types of recommendation approaches in addition to content-based and collaborative filtering. They are demographic, knowledge-based, social filtering, and hybrid systems. Moreover, the connotation of the term recommender system has recently broadened even further (Burke, 2002). Different types of recommender systems have emerged, because the source of knowledge available and other domain factors largely affect building one (Burke and Ramezani, 2011).

In demographic recommender systems suggestions are based on the demographic properties of the user, such as their language or age (Ricci et al., 2015). They are quite simple to implement compared to the other types. Especially the separation of content for different users based on their language or country is already used in many sites.

Knowledge-based systems use experts' domain knowledge to determine how item features meet users needs (Ricci et al., 2015). They are well suited for use in domains with few transactions and relatively static items, for instance, for financial services, real estate, and tourism (Burke and Ramezani, 2011).

Social filtering has become increasingly popular with the growth of social networking sites (Bobadilla et al., 2013). It makes suggestions based on the user's friends' preferences. Also called community-based systems, social filtering is based on the observation that people tend to trust more their friends' recommendations than those of anonymous users (Ricci et al., 2015).

2.5 Hybrid Recommender Systems

Hybrid recommender systems combine different recommendation techniques. Generally the selected techniques are from different types of recommender systems, but it is also possible to combine different techniques related to the

same type (Burke, 2007). Each technique has its own advantages and disadvantages, and they are combined to seek better performance. Pazzani and Billsus (2007) note that strengths of different techniques often complement each other.

The most common hybrid approach is to combine content-based and collaborative filtering (Bobadilla et al., 2013). The strengths and weaknesses of these two types are quite complementary, and therefore combining them helps deal with some limitations, most often the cold-start problem (Adomavicius and Tuzhilin, 2005). Other systems are often content-based or collaborative filtering with additions from other types, like the use of demographic information. This is probably because the research field is not that old and, like Popescul et al. (2001) remind, the first recommender systems were collaborative filtering and content-based.

Different techniques can be combined into a hybrid system in different ways. Examples are, implementing two methods separately and combining their suggestions, and incorporating some characteristics from another method into the existing one, like using content-based user profiles for calculating user similarity in collaborative filtering, (Adomavicius and Tuzhilin, 2005; Burke, 2007). The selected combination affects the complexity of implementation as well as the form of results.

Using a hybrid approach can help reduce data sparsity (Bobadilla et al., 2013), assist in recommending new items or for new users (Burke, 2007), or increase the flexibility and quality of the recommender (Popescul et al., 2001). There are many cases in which hybrid recommender systems have performed better than any single type, but not all hybrid approaches are successful (Burke, 2007). Nevertheless, they are common (Adomavicius and Tuzhilin, 2005) and well-suited especially for domains in which multiple types of data are needed, like suggesting music (Burke and Ramezani, 2011).

2.6 Evaluating Recommender Systems

Evaluating a recommender system can be demanding. Evaluation often refers to measuring the performance of the implemented system (McNee et al., 2006; Schein et al., 2002). According to Ricci et al. (2015), evaluation should be done in different steps during the building of a recommender system. First, the appropriateness of the selected approach, like the data and type of recommender system to be used, needs to be verified for the use case. Then, several calculation algorithms should be compared to find the one with best performance. Finally, parameters are adjusted and the users' acceptance of recommendations is investigated online using controlled online experiments, for instance.

Initial evaluation is often done offline, because online experiments are more expensive and risky (Gunawardana and Shani, 2015). Even though evaluation can be done offline, real user data from online interactions is preferred over simulated data, because human actions are difficult to simulate accurately. The evaluation can be done using historical interaction data which is divided into a training and a test set (Schein et al., 2002). The training data is used for predicting users' interactions which are then compared to the test set to measure performance.

The aim of evaluation is to assess how well the chosen goals, such as accuracy or acceptance, are achieved. Because the purpose of recommender systems is related to user behaviour, and the goals can be multidimensional, grading the system's performance is not simple. The most common performance metric is prediction accuracy (Gunawardana and Shani, 2015). However, like McNee et al. (2006) argue, accuracy does not account for all important aspects and concentrating on increasing accuracy can lead to creating a filter bubble or other unwanted effects. Because of this, other properties, such as coverage, confidence, trust, serendipity, and handling cold-start, have been brought up as measures for success (Ricci et al., 2015).

Parameters of recommender systems are often adjusted using online evaluation, because the goals are related to user behaviour (Ricci et al., 2015). One approach to finding the optimal values for these parameter is to formulate an optimisation problem (Adomavicius and Tuzhilin, 2005). Joachims (2002) uses click-through data for optimisation. Click-through data contains information on what the user clicked on a given list of prioritised items, and is easy to collect in many cases. The optimisation problem to solve is to find the parameter values that maximise user selections along the prioritised suggestions. The optimisation approach can be extended to use with multiple goals that are exact and quantitative (Rodriguez et al., 2012).

Another way to find optimal parameter values is to do controlled online experiments. Controlled experiments are called by multiple names, including A/B tests, randomised experiments, split tests, and Control/Treatment tests (Kohavi et al., 2009). According to Wohlin et al. (2012), controlled experiments are frequently used in software engineering, because they are well suited for evaluating features against each other. A controlled experiment can also be used for evaluating the overall effect of providing suggestions (Kohavi et al., 2009). Moreover, they are especially suitable for assessing users' opinions (Basili, 1996).

In a controlled experiment, users are randomly assigned to two or more variants (Kohavi et al., 2009). One variant acts as a control group, and other variants have one independent variable changed each, while other variables remain unchanged (Basili, 1996). For example, a web page with no suggestions can be the control group for a page showing suggestions provided by a recommender system.

Selecting which attributes to measure when evaluating a recommender system depends on the application domain, used data, and the purpose of the system in that use case. For example, filter bubbles are frowned upon in the news domain. In the next section we review more specifically the properties important in the online news domain.

2.7 Recommending News

News delivery has changed enormously along the growth of the web. Many shortcomings of traditional media can be overcome with the help of the internet (Himelboim and McCreery, 2012). For example, the web provides greater accessibility, which means that online news do not suffer from similar time and space restrictions as traditional media. However, it also means that the users have many more news sources to choose from, including completely new entities, like blogs and news aggregators. Additionally, the online access has increased the requirements for the speed of publishing news (Mitchelstein and Boczkowski, 2009).

The main source of revenue for online news sites is often advertisements (Mitchelstein and Boczkowski, 2009), which means that it is imperative to get users to stay longer and return to the site. Thurman and Schifferes (2012) say that many news publishers see personalisation as a way to increase users' loyalty to the site. Recommender systems have been used for increasing sales in e-commerce by converting browsers into buyers, enabling cross-sell and increasing loyalty (Schafer et al., 1999). These benefits are largely applicable for news delivery as well: converting browsing into more article reads directly increases revenue, and introducing easy access to more interesting articles can increase time spent at the site and therefore revenue.

News articles are items of low value, and the time used for searching for interesting articles can be interpreted as a cost to the user (Ricci et al., 2015). The complexity of a news article is relatively low, but the data is mostly unstructured (Pazzani and Billsus, 2007). Additionally, there often is a massive number of items and the pool of items is in constant change (Burke and Ramezani, 2011). Li et al. (2010) describe the main issue of a news recommender system as assisting the users in finding the most appropriate content at the best time. These aspects lead to many challenges in recommending news.

Because of the time-sensitive nature of the news domain, the cold-start problem is especially consequential (Schein et al., 2002). The item-side cold-start problem is most common in the online news domain, although the other two types can be also encountered. Because of the importance of recommending new news items, a pure collaborative filtering system that suffers from the item cold-start problem is not well-suited for that domain (Lam et al., 2008).

Proper scalability is another requirement in the news domain, due to the immense amount and dynamic nature of data (Ricci et al., 2015). Liu et al. (2010) point out that news sites pursue to present the most recent information. Consequently, the recommender system has to operate fast and in real-time. The large amount of data supports the use of a model-based system, whereas the need for real-time suggestions promotes the use of a memory-based system. Because of these two aspects, both types of approaches face at least some scalability issues.

Filter bubbles are an impediment to news recommendation (Maccatrozzo, 2012). Nguyen et al. (2014) explain the filter bubble as isolation from diverse viewpoints or content. Recommender systems easily provide only content the user is assumed to like (Maccatrozzo, 2012), which can lead to increased revenue in the short run. However, the filter bubble can cause negative cognitive effects on the users (Nguyen et al., 2014), and it can twist the users' opinion of the news site. Moreover, users are usually looking for new, even surprising, information at a news site (Liu et al., 2010). Therefore, recommender systems generally try to avoid creating filter bubbles.

Another challenge in the news domain is the temporal nature of user data (Li et al., 2014). Ricci et al. (2015) propose dividing the user preferences into long-term and short-term. The long-term preferences are usually more stable, whereas short-term preferences are more prone to change (Li et al., 2014). According to Liu et al. (2010), short-term preferences often follow the general news trends. Therefore, taking both the long-term and short-term interests into account could help to more accurately predict the user's current news interest.

Other common issues recommender systems can face are transparency, contextuality, and flexibility (Adomavicius and Tuzhilin, 2005). Transparency means that the users can get explanations for the suggestions (Bobadilla et al., 2013). Transparency helps build trust in users (Ricci et al., 2015), which is valuable for news sites. Contextuality means that the utility of an item to the user may depend on time (Adomavicius and Tuzhilin, 2005), which is often the case in news and is related to aforementioned short-term preferences. Flexibility here refers to the possibility that users can refine the recommendations, which can in turn help users accept the recommendations (Ricci et al., 2015).

Despite the many challenges, recommender systems for news sites have become increasingly popular, and many news providers already exploit this technology. For example, New York Times (Spangher, 2015), Svenska Dagbladet (Rodrigues, 2017), and Neue Zürcher Zeitung (Ciobanu, 2017) recently paid significant attention to recommender systems as a part of their websites. Also, Thurman and Schifferes (2012) point out that there has been consistent growth in using personalisation on news sites.

Chapter 3

Methods

The process of creating a recommender system depends on the application domain: available data and its sparsity, scalability and performance requirements, objectives, desired quality of results, and other possible constraints (Bobadilla et al., 2013). This chapter presents the methods used in building and evaluating the online news recommender system of this thesis.

3.1 Domain and Output

The recommender system is built for a news website for which the speed of providing content for users is crucial. The site is used by millions of unregistered users, and hundreds of articles are published every day. The recommender system generates a list of suggested articles for users. That list is designed to replace the main content on the site's existing front page, unlike in many cases (Ricci et al., 2015), in which the suggestions are shown on a separate page or list.

The news site's current front page design consists of several elements, and the main content is a list of clickable items referring to single articles. Those items are designed to visually attract the users: the article title is often specially

formatted and additional information, such as a picture, time stamp, or a short ingress, is displayed. The visual aspects of the new list of suggestions are as similar to the existing list as possible to help with evaluations (Li et al., 2011) and to increase the acceptance recommendations (Ricci et al., 2015).

In addition to the restrictions arising from the domain and the existing site, the news company also sets constraints for the system, which heavily affect the choice of approach. One popular approach is to use collaborative filtering (Burke and Ramezani, 2011), but because of the cold-start problem, the changing relevance of data, and performance requirements, the choice for approach is content-based.

Existing data consists of only item data: article details such as publication time, section, article links, and various free text fields. Most of the data in manually input by editors. The sections act as classes for articles, to separate completely different articles from each other. Conversely, article links are references to other very similar articles.

In addition to item data, a recommender system requires user and transaction data. To protect user privacy, each user is assigned a random-generated identifier and no personal information is collected. This identifier is used for storing the user's activity which consists of front page load time stamps and read articles.

Only implicit data is collected so as not to effect the users' browsing experience. Front page loads are used as contextual information for interpreting what content the user has probably not seen yet. A read article is interpreted as a positive vote for the article's section like, for instance, in Liu et al. (2010).

A new attribute called *article importance* is formed from existing data. Its purpose is to reflect the DJs' — the online news editors in charge — views of the article, and it is calculated based on the compositions of the front page made by the DJs. Implications of an item's progress on the front page used

for the calculations are assessed by interviewing the DJs. Article importance plays a significant role in calculating the final scores because it also reflects current news trends. It could be given directly by the DJs (Rodrigues, 2017), but that would require more work from them and include less information and possible biases.

3.2 Generating Suggestions

For each user, the suggestions are generated by calculating a numerical score for each article. The final score is calculated for each article based on article importance, freshness, user preferences, and contextual information. The articles are sorted based on this score and the top N items are displayed to the user. In order to prevent forming a filter bubble and to ensure delivery of the most major news, a controlled percentage of items are selected with no regard to user preferences.

The available article data is mostly unstructured, but only structured parts are used for the recommender system. The selected features are the sections and article links defined by editors. This means that no feature extraction method is used. Such a scope is thought to speed up the building process and to enable the approach to be completely memory-based even with strict performance requirements.

In a fast changing environment, such as the news domain, a memory-based approach helps keep the results updated (Bobadilla et al., 2013). Additionally, user data is kept only for four weeks to better account for possible changes in preferences. The user preferences are based on the distribution of read articles in different sections. They are calculated every time suggestions are requested by the user and are based on all transaction data accumulated up to that point Additionally, the data on read articles is used for separate weighting so that it is possible to control how likely already read articles are to appear in the suggestions.

Users are probably interested in the most recent news, but also in some bigger events that have been in the headlines since they last visited. Therefore, their last visit time is used to provide a context for determining what is the most relevant content for the users each time they visit the site. Articles relevant to the context are given more weight in the suggestions. On the other hand, even though some item is major news, if it is likely that the user has seen the item already, this item can be given less weight.

The article importance is used as a basis for scores, but every article does not end up on the front page, and there can sometimes be delay. Therefore, the most recently published articles are also considered worthy for suggesting. Freshness is one aspect that is considered for all articles, because news items usually expire, and often rather quickly. The articles are considered fresh enough if they are currently on the hand-composed front page. Freshness degrades starting from time of removal from the front page, or the time of publication.

3.3 Evaluating the System

The suggestions are provided as main content on the front page, and therefore they require no explicit actions from users to access. This enables making the change without users noticing it (Schafer et al., 1999). This is thought to reduce biases when evaluating the system. In the building and evaluation phase, only a small random subset of users are directed to see the personal recommendations instead of the original front page.

Different algorithms are not compared during the development of the recommender system, even though this is common (Gunawardana and Shani, 2015). With such volatile item data and sparse transaction data, offline evaluation is expensive. However, online evaluation is more expensive and involves some risks. Therefore, only slight modifications to calculations and different parameter values are tested, and the test group size is kept relatively

small. Online evaluation is done with controlled online experiments, which is common in software engineering (Wohlin et al., 2012). Doing controlled experiments is more feasible than finding parameter values by optimisation, because of restrictions for available data and the many goals.

The evaluation of the article importance is done by examining and analysing its behaviour on different articles as a sanity check. Because the article importance is composed based on restrictions from the DJs, the evaluation is partly based on interviews with the DJs. The validity of the article importance reflects also the success of the suggestions.

Sensitivity analysis is performed to see what effects each parameter has on the suggestions. Because the user's context also affects the results, the analysis is done with the data of multiple users that have different behaviour. The analysis is executed by comparing suggestion lists provided for these users with different values of parameters changed individually. The results are fetched at one time, over a minimal time span, to prevent changes in results to be caused by the changes in the data set.

Although offline evaluation is expensive in this case, it is done once to give insight on the accuracy of suggestions. It also enables looking into the operation of the system before acquiring test users. To do the evaluation, all item data and data from a random subset of users is downloaded from a two-day span. This data is divided into sets based on time stamps: for each user, the user's transaction data before a selected time of one article read is used to predict that article based on all other data available at that time. This can be done to multiple articles, up to all articles the user has read. Using time stamps is crucial in such a dynamic setting, and they can be used for examining the accuracy of recommendations as a function of number of read articles.

The results from the sensitivity analysis are used for deciding which parameters and values are selected for controlled online experiments. Besides parameter values, controlled experiments are used for assessing the success of

the system compared to the original front page. Because of some differences in the structure of the main content between the suggestions and the original front page, a so-called placebo option is used as a baseline. The placebo option has the same items as the original page, but displayed with the exact same visual look as the suggestions. The placebo option is compared to the original and the other options are compared to the placebo in order to calculate the success.

The success of the recommender system is measured with five quantitative metrics that describe user behaviour. They include click-through rate (CTR), average number of sessions per user, average number of page views per session, average session length, and bounce rate. CTR is an important measure of advertising effectiveness (Linden et al., 2003), but is not considered to describe user activity sufficiently in this case. On the other hand, sessions per user is considered the most important metric. Relative values for the metrics are used in this thesis to protect business-sensitive information (Li et al., 2010).

Chapter 4

Implementation

As a part of this thesis, a recommender system is built to become a part of a news site. The generated suggestions are made available to users via the existing site. This chapter describes the relation between the recommender system and the existing system, and considers the issues in the implementation of data collection and aggregation, as well as details of calculating scores.

4.1 Architecture and Data

The recommender system is created as a separate component, but is integrated with the current news site system. In the visual illustration of the architecture of Figure 4.1, new components are shown in orange, and existing ones in grey. The recommender system takes a user profile id as an input and provides an ordered list of item ids. Then, other components handle showing the suggestions to the user in a visually pleasing way on the website.

The recommendations page on the site is very similar to the manual front page. The differences are that only one size of items are used, and there are some non-article items as well as additional content at the bottom of

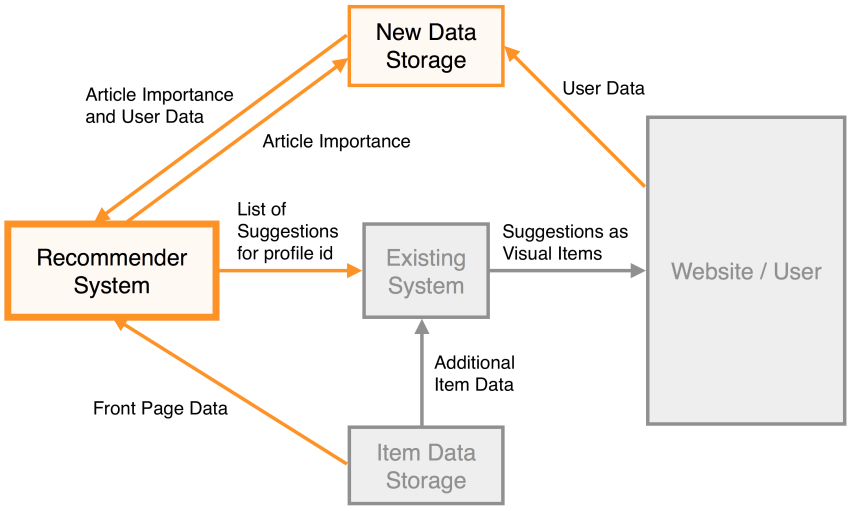


Figure 4.1: Diagram of the system architecture.

the page, which are not displayed in the recommendations page. The visual aspects of the suggestions are not a part of this thesis, but the effect of the differences are taken into account when analysing the results.

The existing system does not provide enough data for making recommendations. Consequently, additional user, transaction, and item data collection is set up. Because the users are not registered, each browser is considered a different user and assigned a random-generated id. Consequently, a person using the news site on different devices is treated as multiple users.

The collected transaction data consists of article reads and front page loads. To reduce noise in that data, all entries are not saved immediately: articles are interpreted as read only after the user has spent number of seconds reading, which can be controlled with a parameter. Additionally, all transaction data is stored for a maximum of one month to decrease total amount of data.

There are some data challenges. For example, an average user reads approximately ten articles, but the median is only two in one month. Additionally, the section distribution of articles is uneven: there are nearly one hundred sections, but there are three big sections that together contain over 40% of the published articles.

Integration to an existing system adds restrictions. The existing system does not support collecting detailed data on user actions, including data on which recommendations the user has seen and which clicked. Also data on where from the user accessed an article cannot be collected. This has a significant impact on evaluating the system.

To easily adjust the behaviour of the system, many parameters are used in determining suggestions. Each parameter has a default value that is used if no modifications are made. Additionally, the value of each parameter can be changed for all users, or groups of users via an interface. This is used for executing controlled online experiments. For sensitivity analysis, it is also possible to examine a single user's suggestions and explore the effects of different parameter values on that list of suggestions.

4.2 Article Importance

In order to help determine which items should be selected for suggestions, a new attribute, article importance, is determined. It represents an editorial view on the articles, and is calculated based on the front pages composed manually by the DJs. The article importance score $I \in \mathbb{R}^+$ of an item depends on the inspected time range $[t_0, t_1]$. Each time range is the time between two manual edits made by the DJs, so the attributes of the items stay constant over the inspected time range.

The article importance score for an article a calculated between two times t_0 and $t_1 > t_0$ is

$$I(a, t_1, t_0) = p(a, t_0)\, s(a, t_0)\, \overline{U}_r(t_1, t_0)\, \Delta t, \tag{4.1}$$

where t_0 is the time of previous edit, $\Delta t = t_1 - t_0$ in minutes, $\overline{U}_r(t_1, t_0) \in [0, 1]$ is the average relative number of users at the examined time period $[t_0, t_1]$, and $p(a, t_0) \in \,]0, 1]$ and $s(a, t_0) \in \mathbb{R}^+$ represent a position score and a size score, respectively.

The importance score is calculated every time the manual front page changes. The scores determined at different points in time are summed to account for the whole period of time the item has spent on the front page. The total accumulated importance for article a at time t is

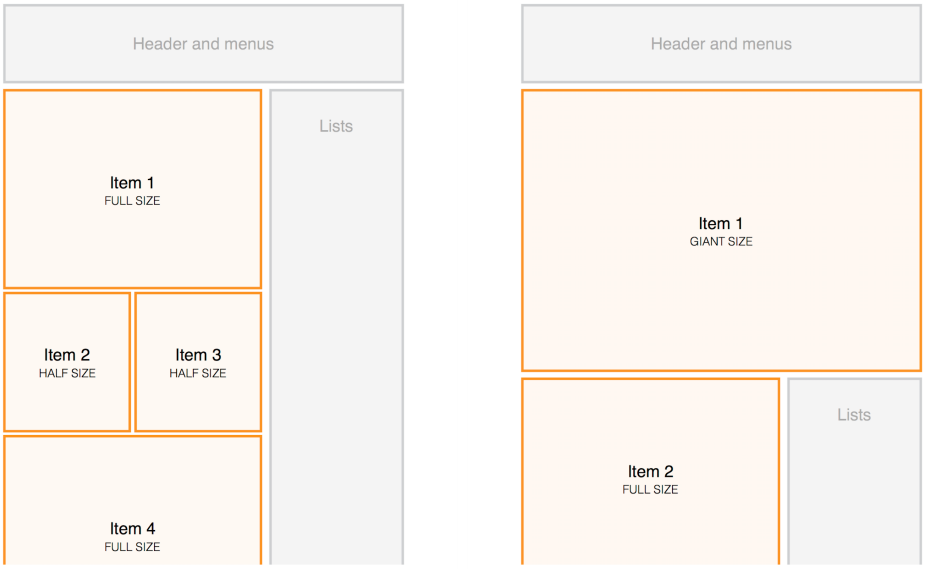
$$\mathcal{I}(a,t) = \sum_{\forall\, [\tau_0,\tau_1]\in T \mid \tau_1<t} I(a,\tau_1,\tau_0), \tag{4.2}$$

where T is the set of all saved time ranges $[t_0, t_1]$ between the front page edit times. Because the importance score is multiplied with Δt, the change frequency will not affect the score. This cumulative sum is saved to a database for use in suggestion calculations and is only article-specific, not user-specific.

There are three different sizes of items: giant, full and half size. Figure 4.2 illustrates the different item sizes and their relation to the layout of the page on big screens. Orange items represent the main content, which is also the target of this thesis, and grey areas represent the other content that is visible to the user. The grey area includes, for instance, automatic lists of most read and most recent articles. On smaller screens, such as on mobile phones, there are no side lists. However, the item sizing is done for the bigger screens, so the focus will be on them.

The item size implications are based on interviews with the DJs. Full size items are used as 'normal' items, and the other sizes are used as means of particular effects. The giant size is reserved for remarkably big news items, and is hence very rare. According to the DJs, the news item needs to be approximately eight times as significant as a usual news item for the giant size to be used. Figure 4.2 shows that the giant size can be only used for the first items of the list because of layout restrictions.

A half size item is given nearly as much importance as a full size item, despite being less easily noticed because of the smaller visual size. Based on the DJs' interviews, the smaller items are used for multiple purposes. They are used for articles with less content, articles on certain topics, and sometimes as visually balancing elements. Based on these properties, the giant, full, and half sizes are given static numerical scores of 8, 1, and 0.85 respectively.



(a) Normal (b) With a giant item

Figure 4.2: General examples of page layouts and different item sizes.

According to the interviews with DJs, position on the front page is the main factor of the article's importance: the higher up an article is, or in other words the smaller the index, the bigger the importance. Additionally, users start browsing from the start of the page and therefore fewer users see the items lower in the page. As a consequence, the position score has to decrease when the index increases.

No similar study for calculating an importance score from a manual front page was found, so the calculations are based on features assessed from the DJs opinions. The shape of the curve is determined by the constraint given by the DJs that position is more important than size. The position score is

$$p(a,t) = c^{n(a,t)-1}, \tag{4.3}$$

where $n \in \mathbb{Z}^+$ is the one based index of the article a in the list of main content at time t, and $c \in]0,1[$ is a parameter describing the decrease in importance when positioned to a bigger index. Because of the constraint that position is more important than size, c must be smaller than the half size score. The

formula includes a normalisation of p to the range $[0, 1]$. The position scores and their relation to sizes with $c = 0.82$ and size weights 1 and 0.85 for full and half size items, respectively, are displayed in Figure 4.3.

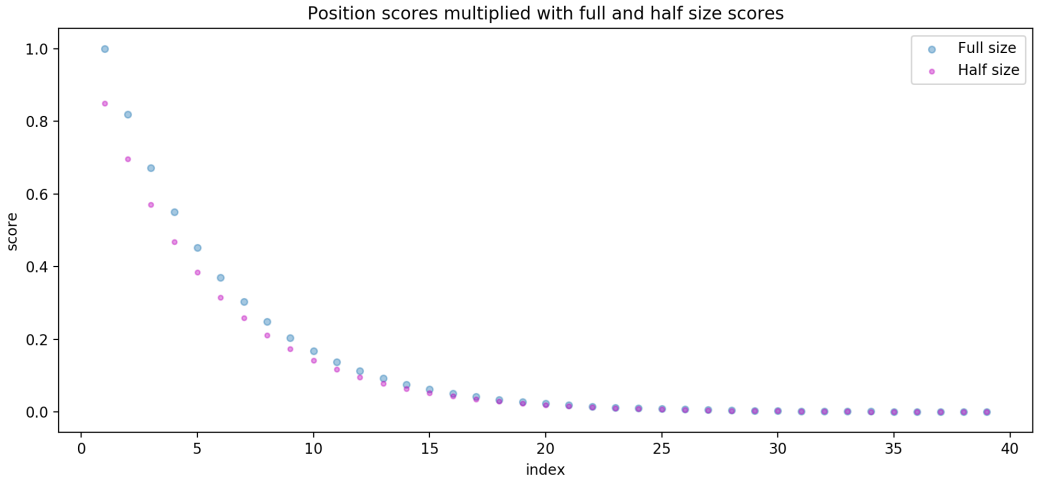


Figure 4.3: Item position and size scores.

In addition to position and size, the time spent on the front page relative to the time of day affects the article importance. The longer an article stays on the front page, the bigger its importance. However, there are fewer users at night, and the DJs controlling the front page know that there is therefore a smaller need for changes. Figure 4.4 displays an example of the temporal changes in the number of users during a four-week period. There is a clear one-week cycle, biggest differences in one day cycles, and no trend.

To reflect the differences of daytime and nighttime, one part of the article importance score is scaling with an average relative number of users. Four months of five-minute average data on the number of users is used for calculating one average week. The five-minute averages are then divided with the maximum to achieve relative numbers. Because there is no trend visible, a static list of averages is used as a reference data of the number of users. The average relative number of users $\overline{U}_r(t_1, t_0)$ is calculated from every five-minute average in the interval $[t_0, t_1]$. Finally, the value is multiplied with the time difference Δt in minutes.

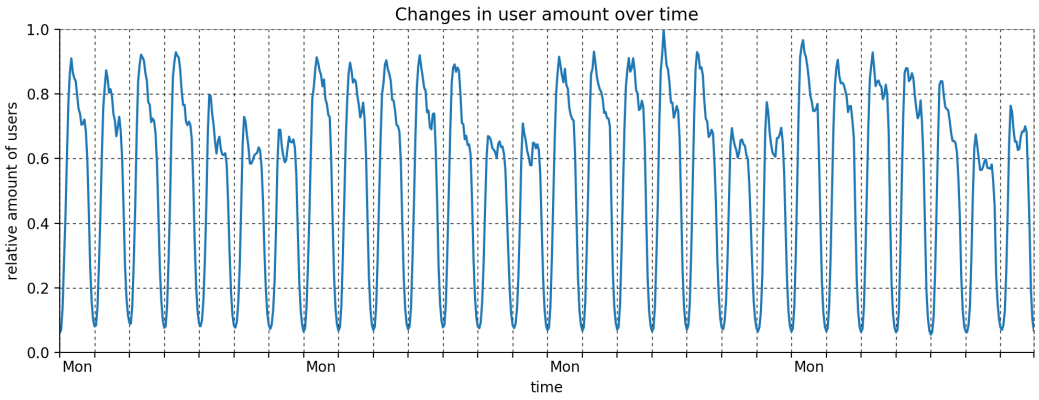


Figure 4.4: Changes in number of users over four weeks.

The summed article importance scores are always positive, and therefore the cumulative sum increases for as long as the article is on the manual front page. Once the article has been removed from the manual front page, its score stays unchanged, and the score is removed from the database after one month to reduce the total amount of data.

An example of behaviour of the manual front page is presented in Figure 4.5. In the figure, each colour represents a different article, each vertical line of markers represents a point when a change happened, and different type of markers represent different item sizes. A star marker represents full size, a dot represents half size, and a octagon represents giant size. Two articles are highlighted with thicker lines as examples, and the point where the size of one of them changes is also highlighted. The plot shows that there is a lot of activity during one day, and the change frequency is not even, but there can be seen a clear decrease in activity during the night.

Not all articles end up on the front page at any point. Even if an article has not been on the front page, it may be worthy of suggesting. For example, articles from certain sections are placed on the front page less often. Articles that have not been on the front page or have just been put on it have $\mathcal{I} = 0$, which results in a final score equal to zero, and therefore those articles are not suggested. To prevent this, the median summed importance is used as a mock importance $\mathcal{J}$ for those articles instead.

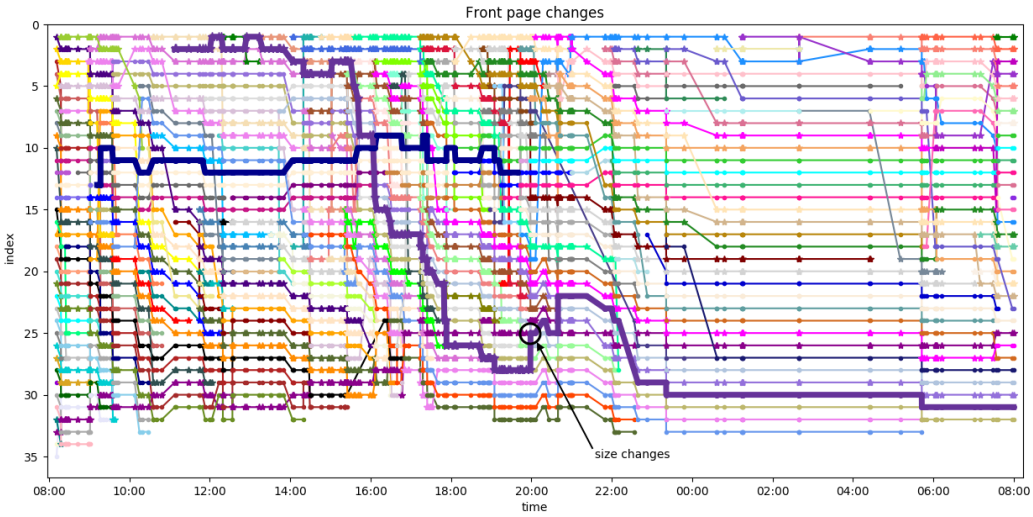


Figure 4.5: Item indices (vertical axis) on the manual front page over one day (horizontal axis).

The cumulative sum of article importance is scaled with a freshness score to promote recent articles, especially those that are on the manual front page at the time of calculation. Because this freshness score depends on the calculation time, it is not saved to the database. When removed from the manual front page, at time t_r, the freshness immediately drops by a percentage, expressed with weight $w_f \in]0, 1]$, and then continues decreasing over a lifespan, parameter $l > 0$. The freshness for article a at time t is

$$F(a, t) = w_f \left(1 - \frac{t - t_r}{l}\right), \quad F \geq 0, \tag{4.4}$$

where the time difference $t - t_r$ and lifespan l are in minutes. Articles that have not been on the front page have $w_f = 1$ and the comparison time stamp t_r is the article's publication time.

4.3 User Preferences

The user's u preferences $P(u, a)$ are derived from the section distribution of the user's read articles:

$$D(u) = \left(\frac{N_1}{N_{total}}, \frac{N_2}{N_{total}}, \ldots, \frac{N_n}{N_{total}} \right), \tag{4.5}$$

where N_i is the number of read articles from section $i \in \{1, \ldots, n\}$, and $N_{total} = \max(M, \sum_i N_i)$. In order not to limit the diversity of suggestions when the user read very few articles, a parameter $M > 0$ is used as minimum number of read articles. In that case D is not an actual distribution, because the weights sum to less than one.

Two different approaches for modelling the user preferences are compared. A parameter is set for changing between the two approaches for easy comparison of the differences in results. The first approach is to set the preference scores P equal to the section distribution D. To be able to suggest articles from sections outside the user's preferences, a mock preference score Q is used for those articles. The mock preference is equal to what the preference would be if the user read one article from that section. Thus, the mock preference is always smaller than the actual preferences, but only by little.

The users are likely to read more articles from the big sections, because there is more articles available. As a consequence, big sections get often big weights with the first approach. Another approach is chosen to reduce the effect of sections size on the preferences. The other approach uses binary preference scores. For each section that has $D > 0$, the preference is set equal to a fixed parameter $w_s > 0$, and the preference for other sections remain is set to a significantly smaller $w_{s0} > 0$. The w_{s0} is then used also as the mock preference score Q.

4.4 Other User Scores

The articles read by the user are also taken into account separately. A read article score is controlled with a parameter $w_r \in [0, 1]$

$$R(u, a) = \begin{cases} w_r, & \text{if article } a \text{ has been read by the user} \\ 1, & \text{otherwise.} \end{cases} \tag{4.6}$$

If the parameter w_r is set to 0, read articles can not appear in the suggestions, and if set to 1 they have no effect on the final score. When the parameter is set to something in between, read articles drop to lower positions in the suggestions and therefore may or may not end up in the suggestion list because of its limited size.

To account for users with different visit frequencies, the score is also affected by the user's last visit time. This is represented by the temporal context score defined by a parameter $w_c \in [0, 1]$

$$C(u, a) = \begin{cases} 1, & \text{if article } a \text{ is considered recent} \\ w_c, & \text{otherwise.} \end{cases} \tag{4.7}$$

Here recent means that the article has either been put on the manual front page or published after the user's previous visit. These articles are more likely not to have been seen by the user and therefore have a higher final score. In order to separate the user's current session from the previous one, a minimum time span for the context is set with a parameter T_{min}.

4.5 Generating Recommendations

To form recommendations for a user, a final score is calculated for each relevant article. An article is considered relevant if it was published or has been on the front page during the past few days. The exact time limit is controlled by the parameter l which is initialised to one day. The final score

determines the set and order of suggested articles. The final score S of article a for user u at time t depends on the article's importance $\mathcal{I}$ and freshness F, and the user's preferences P, read articles R and temporal context C.

$$S(u, a, t) = \mathcal{I}(a, t)\, F(a, t)\, P(u, a)\, R(u, a)\, C(u, a). \tag{4.8}$$

Multiplication is used because the sum of importance scores is not bound from above and the rest of the terms act only as weights. A linear model would make analysis easier, but would require re-thinking the behaviour of the article importance scores' relation to the other parts of the final score. The different parts of the final score are presented in the following sections.

Some adjustments to the calculations and the suggestion list are needed to provide fresh and diverse suggestions. Often there are multiple articles on one news event: a new article is published when new information regarding that event is discovered. The content of these articles often overlaps quite significantly, and it is not preferred to show too similar articles. Fortunately, these articles are usually linked to each other by the editors. Therefore, to prevent suggesting too similar items, the system filters articles that are linked to other articles in the suggestions.

Ensuring the diversity of suggestions is vital, especially when the user has few read articles, in order to prevent the forming of a filter bubble. This is done by selecting a certain percentage of articles, called the bubble breaking items w_b, without regarding the user's preferences. First the article scores are calculated without the user preference term P, and the bubble breaking items will be those with the highest scores. These items are not the same for all users, because the scores still depend on the user-specific terms R and C.

After the bubble breaking items have been selected, the remaining items are selected based on the final score. However, to enable providing suggestions from small sections, the results are limited section-wise with the distribution D. Items with the biggest scores within the same section are selected from each preferred section. As a result, the section distribution of the articles suggested is in proportion to the user's section distribution.

Because the recommender system is going to replace the main content on the front page, an additional restriction, called fixed items, was given by the DJs. Fixed items are articles that are put to the top of the suggestions directly from the current front page. They are the same for all users. The number of these items is controlled by a parameter n_F, which is initialised to 3. Additionally, the parameter enables the creation of the placebo suggestions page that is article-wise the same as the manual front page by setting $n_F = n$, where n is the total number of suggested items.

Chapter 5

Experiments and Results

This chapter presents the experiments, and the evaluation results for the performance metrics. The impact of the different parameters presented in the previous chapter is studied. The appropriate parameters are selected for controlled online experiments, to measure the success of the recommender system.

5.1 Article Importance

Article importance is significant in the suggestion calculations. However, its accuracy is difficult to evaluate, because it is based on the views of the DJs. Therefore, the evaluations of article importance are based on examining the behaviour of the score for different kind of articles and reflecting to the DJs' opinions. In addition to article importance, the forming of the freshness score is analysed and evaluated in this section, because those two are very closely related.

Figure 5.1 presents the corresponding article importance scores calculated with Equation 4.1 from the activity presented in Figure 4.5. In these calculations, the position factor is set to $c = 0.82$, and the size weights for half,

full, and giant item sizes are 0.85, 1, and 8, respectively. Each line in the plot represents a different article, and the same articles are highlighted as in Figure 4.5.

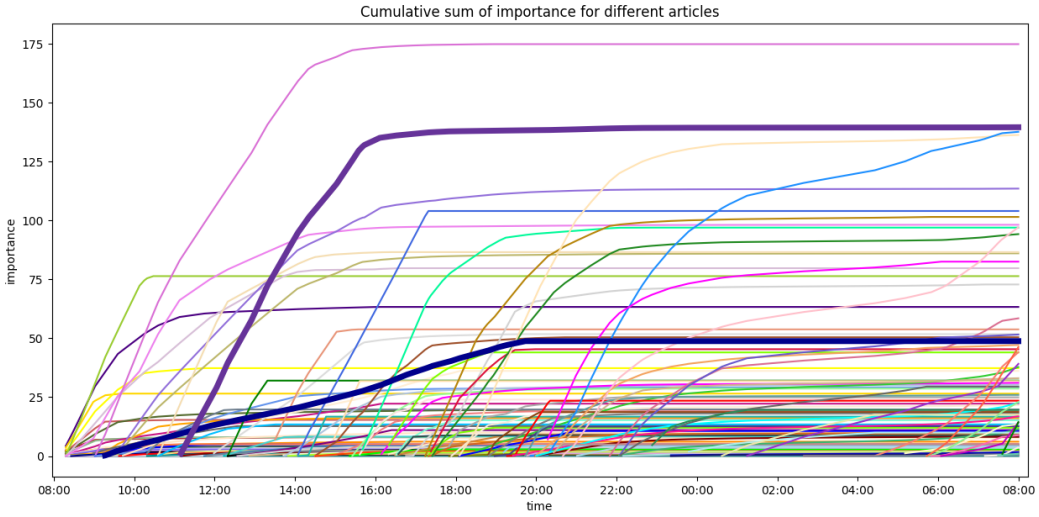


Figure 5.1: Article importance score development for articles over time.

In Figure 5.1, all article importance scores start from 0 at the start time 8:00, because no earlier data is regarded in these calculations. This reflects a situation in which the system has only just started running, but it affects only the scores of those articles that were on the front page at the start time. Even if those articles that have incomplete scores are not regarded, the majority of the articles have a relatively small score of 25 or less; the density of lines in the figure increases towards the score 0.

The curves of the importance scores vary largely. For any article, the increase between the manual front page changes made by DJs is linear, and the slope changes only if the change affects that article. For instance, the highlighted blue article stays approximately in the same position, same size and during a time of day when the user amount stays relatively constant for the first 10 hours. Therefore there is no clear change in its score's slope. On the other hand, the highlighted purple article experiences many changes during its time on the front page, which result in a less linear overall score curve.

The slopes in Figure 5.1 are steeper during the day than the night because of the relative user amount term in the score. For example, the highlighted purple article is first high up in the list during the day, so its score increases fast. Then, in the evening, it has a low position and its size is smaller, resulting near-zero increases in article importance score.

There are no giant items of articles in Figure 4.5. The giant size is reserved for more rare, major news items, and it has a big weight in order to gain a big score fast. Comparing to the biggest score in Figure 5.1, for instance, a giant item would pass that score it in less than half a hour on a weekday morning. Conversely, during a quiet weekend night, a giant item would pass that score in approximately four hours.

An important feature to notice from Figure 5.1 is that all the curves are monotonically increasing. As long as an item stays on the front page, its score increases. After removal from the front page, the saved article importance score remains constant. The relevance of a news article decreases over time. The relevance is thought to start decreasing after the article has been removed from the front page. It is handled by a freshness score, which is calculated on-to-go as it depends on the calculation time.

Figure 5.2 displays the effect of scaling the importance scores of Figure 5.1 with freshness (Equation 4.4). The parameters used for the freshness score in the figure are $w_f = 0.5$ and lifespan l equal to one day. The freshness scores here are calculated at the same save points as the importance scores. In this figure, the time of removal from the front page can be seen as a big drop in score. An article's freshness decreases immediately when it is removed from the front page to promote more recent articles and those chosen by the DJs.

In Figure 5.2, the article highlighted with purple becomes the one with the biggest score when the previously most important is removed from the front page. The purple article remains on the front page, so the DJs consider the article still relevant to the users. Therefore its importance score keeps increasing, even though very slightly.

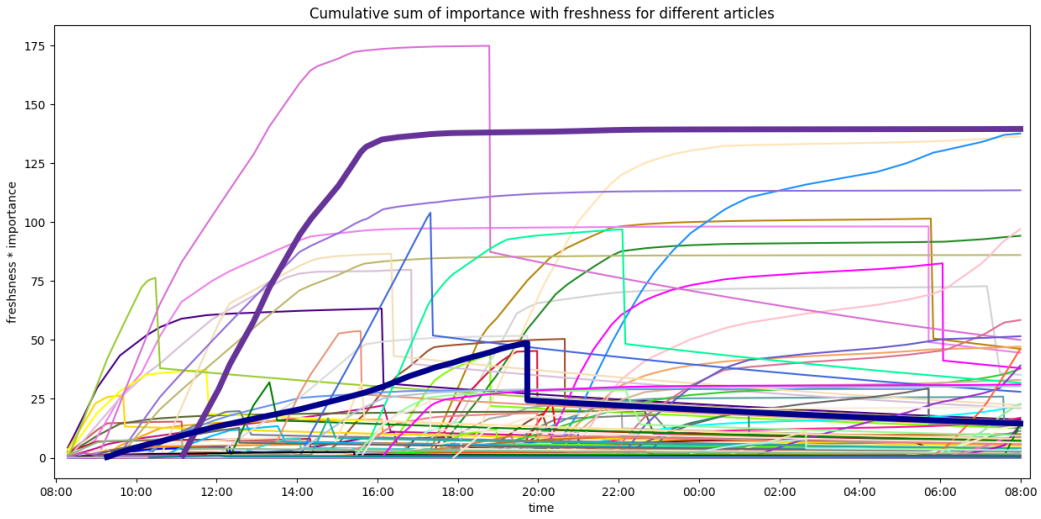


Figure 5.2: Article importance scaled with freshness

When constructing the freshness score, a different approach was first tried. The first approach was a freshness score inversely proportional to the time elapsed from the removal from the front page: $\frac{1}{t-t_a}$. The approach was evaluated by visualising the effects on different articles.

The effects were then compared to the opinions on the articles' lifespan behaviour of the people from the media company. The drop in the score with this approach was considered too big and fast. It was also identified, that it was preferred that the freshness-scaled importance score reaches zero. This approach could not satisfy those restrictions even if scaled of with other small modifications, so it was rejected.

With the new insight on articles' lifespans behaviour, a linear approach was selected. Furthermore, in order give emphasis on articles that are on the front page or have just been published, the weight w_f was introduced.

As presented evaluations of article importance and freshness are only qualitative, small adjustments of the parameters could not be evaluated precisely. However, the importance and freshness scores form the basis for the suggestions, which in turn can be evaluated more precisely.

5.2 Sensitivity Analysis

The system contains many adjustable parameters that have an effect on the suggestions. Sensitivity analysis is performed to see how big the effects are, and therefore to decide on what to test with the more expensive controlled online experiments. Some parameters' effect depends on the characteristics of a user's transaction data.

Because of differences between users, the sensitivity analysis was done with six user profiles. Those profiles were selected randomly, discarding too similar profiles, until the selected set had varying characteristics. The inspected characteristics were the number of read articles N, the section sizes and section distribution of the read articles, visit frequency, and the time of previous visit compared to the time of analysis. These characteristics were selected because the suggestions are based on those features. The selected profiles and their characteristics are presented in Table 5.1.

Table 5.1: Characteristics of selected profiles for sensitivity analysis.

	N	**section distribution**	**visit frequency**	**latest visit**
1	1	from a big section	none	none
2	41	most from a single smaller section	two times a day, not every day	3 hours ago
3	63	many sections, but majority from big sections	once a day	1 day ago
4	35	one third from a big section, one third from a single smaller section, not many sections	same time once every day	1.1 days ago
5	9	both big and small sections, no clear preference	multiple visits on one day	5 days ago
6	63	many sections, similar to the distribution of all published articles	many visits a day	1.5 days ago

Eight parameters, with four to six values each, were selected for the analysis. They are in Table 5.2. For each user, the suggestion list was retrieved with the different parameter values over as short time as possible to minimise changes in item data affecting the results. The results of the sensitivity analysis are presented as generated suggestions with different parameter values in Figures 5.3 and 5.4.

Table 5.2: Parameter values selected for sensitivity analysis.

parameter	**description**	**values**
w_b	bubble breaking	0, 0.01, 0.05, 0.1
l	lifespan	720, 1440, 2880, 5760
M	minimum read articles	2, 5, 10, 100
T_{min}	minimum context	1, 5, 10, 30, 60, 120
w_r	read weight	0, 0.01, 0.1, 0.5, 1
w_f	freshness weight	0.01, 0.1, 0.5, 1
w_s	preference weight	0.2, 1.0, 5.0, 10.0
w_c	context weight	1, 0.5, 0.33, 0.2

In the plots of Figure 5.3, each colour represents a different article. The default value for each parameter is highlighted with a vertical grey line. The changes in the order or set of the suggested articles are of interest in these plots. The shapes of plots for profiles 2, 3, and 4 are very similar to the plots for profile 6.

Despite the similarity of shape of many of these plots, the set of articles clearly varies between the profiles. This means that the system is able to provide different suggestions for different profiles at the same time point. On the other hand, two of the first three suggested articles in the plots for different profiles are the same. This means that the importance score affects the suggestions as intended.

According to Figure 5.3, the biggest changes occur with parameters l and w_f. Other parameters have relatively small effects on the order of the articles. The parameter T_{min} noticeably affects only the suggestions for profile 1. Pro-

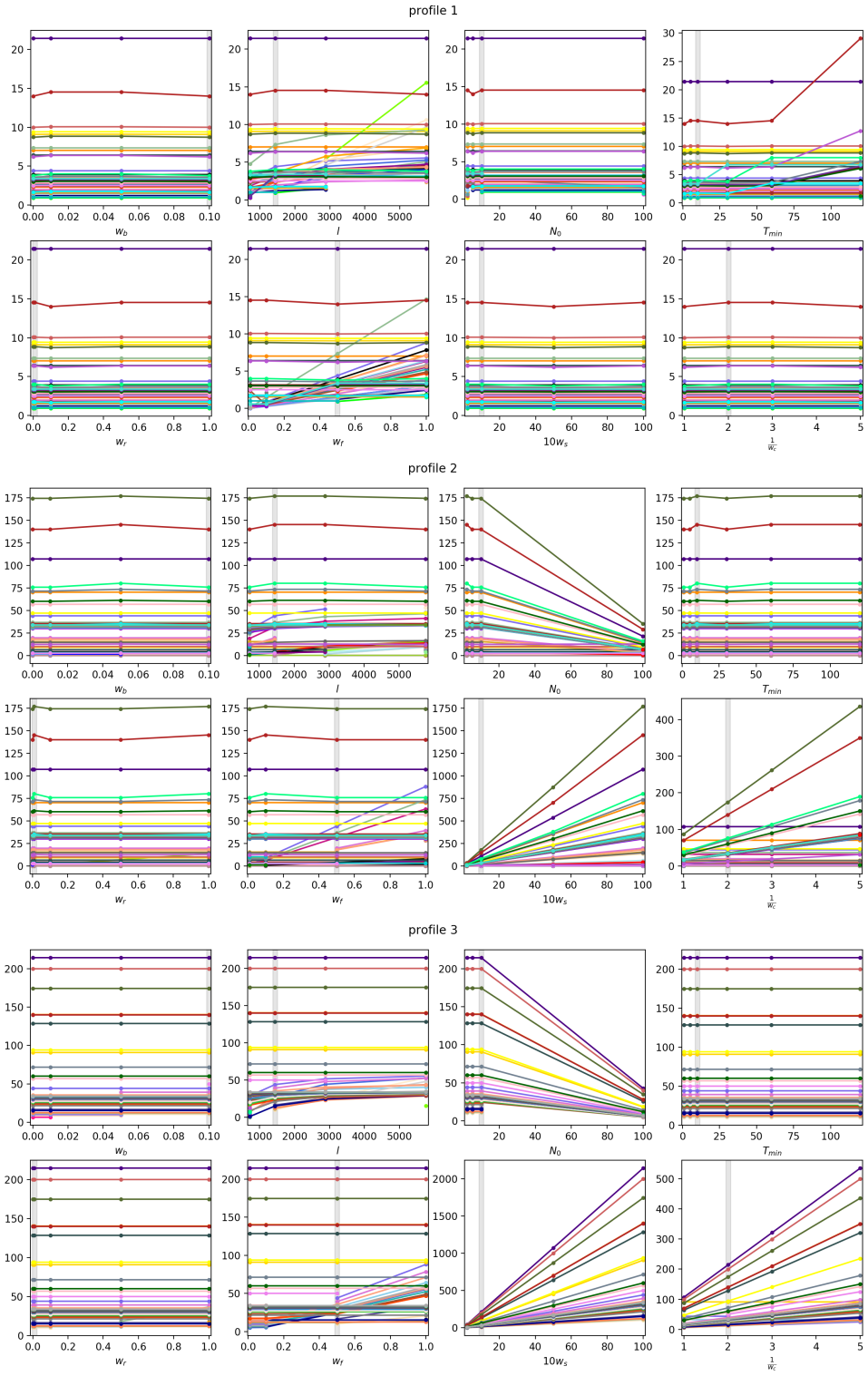


Figure 5.3: Sensitivity analysis results for profiles 1–3.

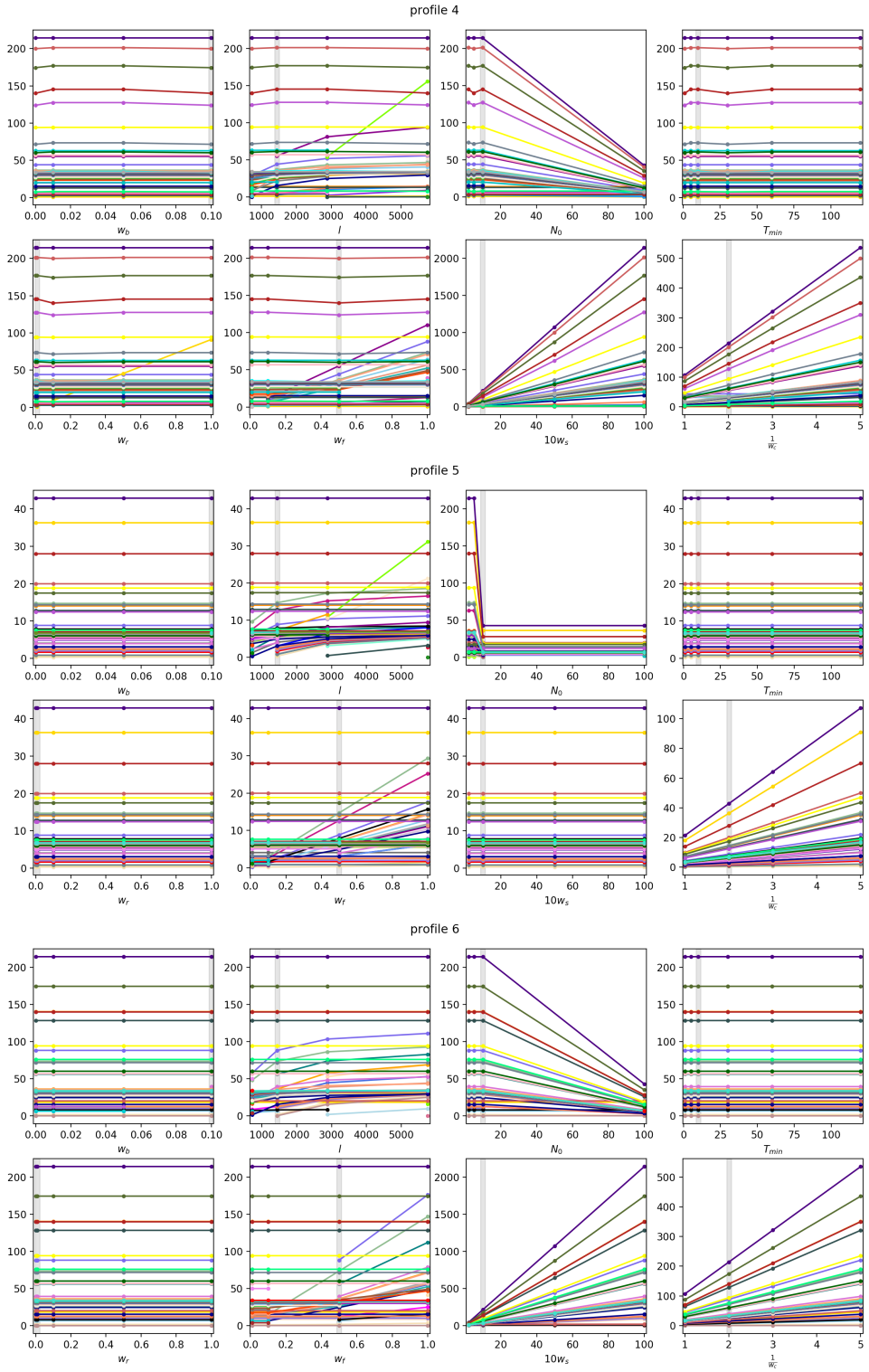


Figure 5.4: Sensitivity analysis results for profiles 4–6.

files 1 and 5 have fewer read articles, and therefore the plots for parameters M and w_s are different from the others. However, even then there are not big changes in the suggestion list structure.

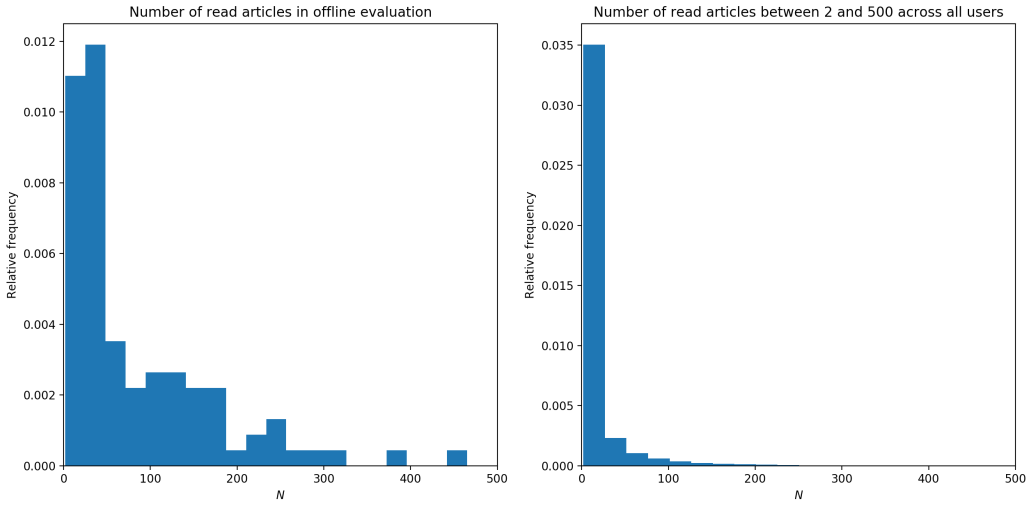
As a conclusion, parameters w_b, w_r, and w_c can be left out from controlled experiments. On the other hand, parameters l and w_f are of most interest. It is not necessary to test the remaining three parameters, but it is possible that they cause changes in the results.

5.3 Offline Evaluation

Offline evaluation was done for 98 randomly selected user profiles with two or more read articles. The selected users had not interacted with the recommender system. One article is selected to be predicted and a minimum one article is required to generate suggestions, so at least two reads are needed per user. Because only two days of item data can be downloaded, the selected users also need to have at least one read within the previous day.

Histograms of number of read articles are displayed in Figure 5.5. Figure 5.5a displays the number distribution of number of read articles of the profiles selected for offline testing. Figure 5.5b displays the distribution across all applicable profiles, with the exception that it is cut at 500 reads for easier reading. The tail is extremely thin, and spans up to over 5000 articles. According to the number of read articles, the selected profiles represent all profiles quite well.

For each user profile, the last read article was picked to form a test set. Then, a suggestion list was created at the time of the user's last read article based on other transaction and item data calculated backwards to correspond the data available at that time. Because the data available at each point in time is calculated from the downloaded data, it is more prone for errors than if the data had been downloaded separately at each time stamp.



(a) Offline evaluation. (b) All applicable profiles.

Figure 5.5: Distributions of number of read articles of user profiles.

Each article in the test set was then compared to the 50 generated suggestions. The distribution of indices of the test articles in the suggestions list are displayed in Figure 5.6. The figure shows that the test article was found in the generated suggestions only in 32% of the cases. Additionally, the distribution of indices in those cases is nearly uniform.

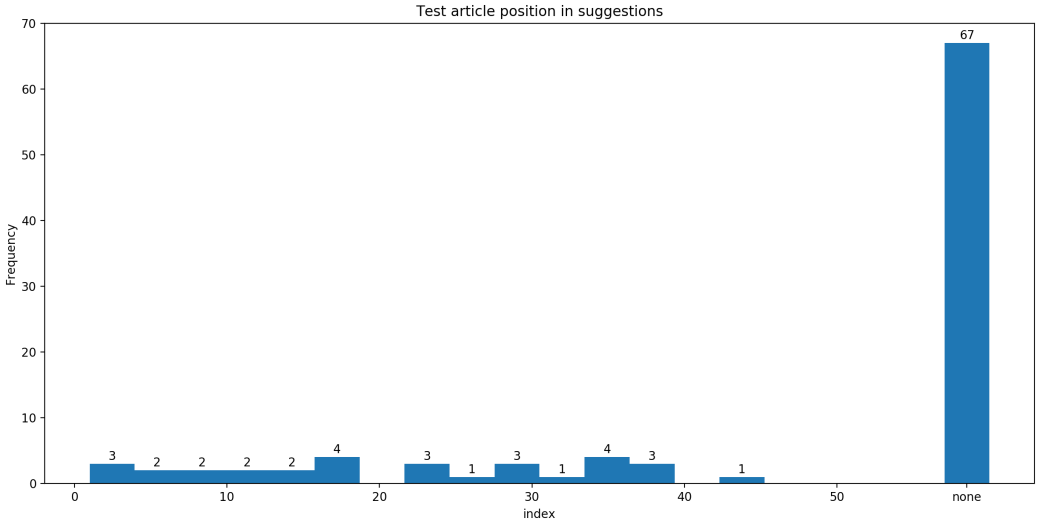


Figure 5.6: Offline evaluation test article indices histogram.

For comparison, also the test article indices on the corresponding manual front page were studied. The distribution of indices is presented in Figure 5.7. For 54% of the users the test article was found on the manual front page at the time. The amount is significantly higher than for the suggestions, but even then the percentage is relatively low. Surprisingly, like in Figure 5.6, the distribution of the indices is also nearly uniform.

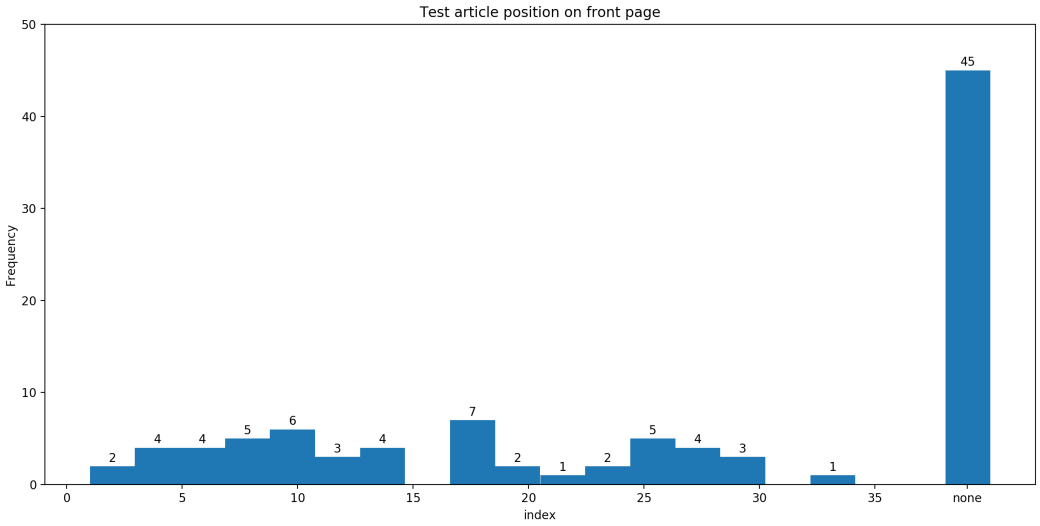


Figure 5.7: Offline evaluation comparison results histogram.

According to the offline evaluation, the accuracy of the recommendations is quite low. However, the articles the user sees at any one time has an effect on what they select for reading, but that data was unfortunately not available. The accuracy gives insight into the performance of the recommender system, but the final success is measured with other metrics.

5.4 Controlled Online Experiments

Controlled experiments are done online to measure the effect on user activity. They are used for both comparing the performance with different parameter values as well as overall performance compared to the manual front page.

First, a small piloting experiment was performed, in which a random subset of users are given the option to join to see their suggestions page. Verbal feedback from these users was gathered to assess the qualitative aspects of success. The number of responses was extremely low, which is common with explicit data. They showed only that, despite the differences other than the order and items of the article list, the users noticed hardly any difference compared to the DJs' front page.

After the first experiment, controlled online experiments with different variants were run. The users were selected randomly from the set of active users. However, for these controlled experiments, the users were not informed that they are participating in the experiment. The purpose of these measures is to make the results as reliable and comparable to the manual front page as possible.

The manual front page is slightly different from the suggestion page, as mentioned in Section 4.1. Therefore, a placebo variant is compared with the manual front page and the suggestion variants are compared with the placebo. The placebo has settings $n_F = n = 30$ and $w_r = 1$ in order to match the content and behaviour of the manual front page as closely as possible.

Parameter values for the default variant are selected based on manual testing and experimentation and their purpose is to form a baseline to compare to. One parameter value is changed for each of the other variants to better examine their effect. The variants were selected based on the results of the sensitivity analysis, and requests from the news agency. The variants are in Table 5.3.

As seen in Table 5.3, a total of 11 variants were run against the normal front page. All of these variants were run simultaneously in order to achieve comparable results. A few hundred users were redirected to each variant without their knowledge. They were not able to access the manual front page during the experiment. The number of users for each variant is relatively low to minimise the cost of controlled online experiments.

Table 5.3: Variants for controlled online experiments.

variant	description
0	normal front page
1	placebo
2	default
3	more weight on user preferences: bigger w_s
4	user preferences: first approach
5	no fixed articles: $n_F = 0$
6	shorter lifespan: $l = 0.5$ days
7	longer lifespan: $l = 2$ days
8	decrease freshness faster: $w_f = 0.1$
9	more results: $n = 100$
10	do not drop read articles: $w_r = 1$
11	longer minimum context time: $T_{min} = 60$ minutes

The controlled experiments were run for one week, because the user behaviour varies between the weekdays and the weekend. The length of the experiment is relatively short, because of the high cost and time constraints. The time period may not be enough to attract new users, for instance. Therefore, the results are not likely to account for all possible changes.

The results of the controlled experiments are in Tables 5.4 and 5.5. Sessions per user is considered the most important metric, but other relations of the metrics are not definite. The bounce rate is practically the converse of CTR, and its values here are reversed for easier comparison. Other metrics are pages per session and session length. For all values in these tables, a larger value corresponds to better performance. Changes in these values were examined during the experiment. Based on the size of the observed fluctuation, the error range is assumed to be ± 2 percentage points.

The relative changes in different metrics compared to the default suggestions variant are in Table 5.4. Variant 8 has significant, positive values for all metrics, so it clearly dominates the default variant. Conversely, variant

Table 5.4: Results of controlled experiments compared to variant 2 (default).

variant	**CTR**	**sessions**	**pages**	**length**	**bounce**(−)
3	−1.17%	18.03%	0.25%	−1.08%	0.80%
4	0.48%	17.10%	−3.51%	0.00%	3.52%
5	−4.44%	14.99%	−4.01%	−9.98%	0.90%
6	−0.29%	15.07%	−1.50%	−2.82%	2.23%
7	−3.96%	26.15%	0.00%	4.12%	1.93%
8	4.28%	9.37%	7.02%	3.47%	5.98%
9	0.87%	14.21%	3.26%	−2.60%	4.52%
10	−3.16%	24.51%	−7.77%	−0.65%	−3.26%
11	0.11%	−0.62%	0.75%	−0.22%	0.60%

11 introduces no significant change compared to the default. The value of sessions per user increases with variant 3, and the other metrics have no significant change. Therefore, also variant 3 can be considered better than the default variant.

There is a substantial increase in sessions per user for all the other variants, 4–7, 9, and 10, in Table 5.4. However, for each one of them, the value of at least one metric decreases significantly. Therefore, none of them can directly be considered better than the default. The increases in sessions per users are even larger than with variant 8, so variant 8 does not dominate these other variants. At least the parameters w_r and l should be investigated further, because the variants 7 and 10 have the biggest positive change in sessions per user.

Table 5.5 has the results of the controlled online experiments for performance evaluation. The manual front page, variant 0, performs poorly compared to the placebo variant 1. Especially changes in sessions per user and session length are huge. Nonetheless, the size of these changes can be explained with the remaining differences in the page elements, like the smaller number of advertisements.

Table 5.5: Results of controlled experiments compared to variant 1 (placebo).

variant	**CTR**	**sessions**	**pages**	**length**	**bounce**(−)
0	1.15%	−65.32%	−8.53%	−35.49%	1.37%
2	0.96%	−13.56%	−8.06%	−23.55%	−3.26%
3	−0.23%	2.02%	−7.83%	−24.38%	−2.44%
4	1.44%	1.21%	−11.29%	−23.55%	0.38%
5	−3.53%	−0.61%	−11.75%	−31.18%	−2.33%
6	0.66%	−0.54%	−9.45%	−25.70%	−0.96%
7	−3.05%	9.04%	−8.06%	−20.40%	−1.27%
8	5.28%	−5.47%	−1.61%	−20.90%	2.92%
9	1.83%	−1.28%	−5.07%	−25.54%	1.41%
10	−2.24%	7.62%	−15.21%	−24.05%	−6.62%
11	1.07%	−14.10%	−7.37%	−23.71%	−2.64%

The placebo variant performs mostly better than the normal front page, but none of the parameter variants dominate the placebo. Most metrics yield a negative change on all the variants. Only variant 8 has slightly positive relative CTR and bounce rate, and variants 7 and 10 have increased sessions per user. Still, they have bigger negative change in other metrics. Therefore, we cannot say definitely that the recommender system increases user activity.

Chapter 6

Discussion

This chapter discusses the success of this thesis. In addition to analysis of the meaning of the results, this chapter includes critical assessment of the methods and results. Finally, possibilities for further work are presented.

6.1 Article Importance and Performance

The main research question was that does the built recommender system increase user activity. Results show that the recommender system provides different suggestions for different users as intended. However, according to the results from the decided performance metrics, there is no clear increase in user activity. Because of no definition of how the different metrics are weighted in assessing the total performance, it cannot be said that the user activity decreased either.

Weights for the performance metrics are required to provide precise results. However, these metrics mostly measure increased revenue, which is not the only success factor. They cannot capture the subjective feelings of the users, which are needed to assess the goal of keeping the general feel of the site.

The short qualitative experiment results showed that the users noticed no difference between the suggestions page and the manual front page. This can be interpreted both as a positive and a negative result. The results are positive when considering keeping the general feel of the site. The negative interpretation is that there was no significant increase in the number of interesting articles in the suggestions.

More controlled online experiments are needed to find the best parameter values. The experiments run in the scope of this thesis only include changing one parameter at a time. Changing multiple parameters together can yield unexpected results. Therefore, parameter combinations need separate controlled online experiments. However, even the best combination may not perform better than the manual front page, at least in short-term.

According to the performance results, the DJs are very good at composing the front page. They are able to account for different aspects than a computer algorithm. This indicates that constructing the article importance score based on the DJs front page is valuable.

The other research question concerned the article importance attribute. It is difficult to precisely evaluate the article importance, but because its purpose is to reflect the DJs opinions, the evaluation done should be sufficient. Even though the article importance score serves its purpose, the purpose may not be entirely suitable for achieving good overall performance.

There is no earlier research similar to the article importance presented in this thesis. In the literature, other methods are used for acquiring and constructing the needed item information. The probable reason for this is that every recommender system environment is unique, so within the news domain there may not be many cases in which this kind of approach would even have been possible.

Many successful recommender systems have been implemented in the news domain. However, often the success has been measured with a single metric, like accuracy or CTR. For comparison, the recommender system presented in

this thesis can be considered successful according to some of the single metrics, and unsuccessful according to others. The total performance assessment is more complex than in many cases found in the literature.

Moreover, this recommender system does not use extensively any of the methods and techniques present in the literature. The recommender system is mostly content-based, but the similarity values are human-provided. No topic modelling or similar technique is used, unlike in many earlier successful recommender systems. This could be one of the reasons behind the poor results.

The success of a content-based recommender system is said to depend heavily on the goodness of the user profile. This can be the main reason for the low success in this case. The section-based user profile does not seem to distinguish preferences well enough. There is no additional separation between different kind of topics and events within the big sections. Therefore, for example, everything between weather and politics can be found in one of the big sections.

In addition to separation issue of the user profiles, the use of section information and article links cause other problems. They are both manually set by the many editors. Therefore, not all articles are in correct sections, and some may not even fit nicely to any of the sections. Additionally, because of the manual work, there are deficiencies and differences in the article links.

There are factors that cannot be controlled, but which affect the results. The time of day, weekday, and the state of the world at the time of gathering results have an effect on the results. This is caused by the rapidly changing and unpredictable nature of the news domain. The effect reaches all the different evaluations done: article importance, sensitivity analysis, offline evaluation and controlled online experiments.

The success metric values are provided by a third party. Therefore, there is little control over them, and neither the reliability nor error ranges cannot be confirmed. This could be changed, however, with a lot of extra work.

Time is one of the biggest constraints limiting the gathering of results. In addition to running more controlled online experiments, more qualitative feedback from the users would be useful in further assessing the success of the recommender system. Both kind of experiments require more time, and should be run for a longer period of time to achieve results of possible long-term changes. For example, the popularity of the site — related to the goal of better catering for different kind of users — is unlikely to change significantly over the course of one week.

Another limitation is that the users are unregistered. There is no way to link different profiles achieved, for instance on different devices, to the same user. This also results in a larger total number of profiles. Similarly, the link between a user and their profile is more prone to disappear. For example, the user may go on a vacation and not use the site for weeks, resulting in losing most or all the acquired transaction data because of the expiration limit. Therefore, using a memory-based approach with user preferences that are fast to calculate in real-time, is more justified.

Collecting more data on the user behaviour could help evaluate and improve the performance of the recommender system. That includes, for example, information on articles the user has seen but has not read, or information on where the user has found each read article. That information is not available, because of constraints from the system that forms the news site.

6.2 Future Considerations

There are many ways to improve the implemented recommender system. Future considerations are given special attention because the system did not achieve wanted results. The ideas presented in this section are relatively small. Bigger structural changes, like incorporating other methods, such as collaborative filtering, could also improve the results, but are not discussed here.

Instead of assessing article importance from the DJs' manual front page, the editors could set certain feature values for each article from which the scores would then be calculated. This would diminish the issue of the importance having a delay. On the other hand, this approach would lead to the DJs jobs to drastically change or even disappear.

Similarly, the editors could set a lifespan separately for each article. Because of the differences in interest period on different kind of articles, this would be a direct improvement over the currently constant-valued lifespan. Moreover, it would not have a big effect on any existing jobs. Similar results could also be achieved from predicting the articles lifespans from the manual front page, but they would suffer from the same delay as the article importance.

The modelling of user preferences could be improved with different elements. Many papers mention the use of temporal user models, i.e. modelling short-term and long-term preferences separately. However, because the user data has an expiration time in this case, there is less concern over changing preferences. Additionally, user data is quite volatile so that storing long-term preferences may not be useful.

User preferences could be improved by making them relative. Instead of using the user's absolute section distribution for preferences, it would be scaled with the section distribution of the general public. This would make the suggestions more personal and help reduce the dominance of the bigger sections. Additionally, the relative preferences could be further scaled with the current news trends. That would ensure that most time-sensitive news end up in the suggestions. It would be a relatively easy feature to add together with the relative user preferences.

The sections are in a hierarchy, which is currently not considered in the user preferences. There is likely to be correlation between the sections close to each other in the hierarchy. This could be used for broadening the user preferences. However, the hierarchy is quite uneven, which makes automated use of it challenging.

A more comprehensive solution for the issues rising from the section-based user preferences would be, for example, topic modelling. Classifying the articles into new topics based on new, extracted features could result in more evenly sized classes than the sections. Topic modelling could also be used in a smaller scale to create additional classes within the big sections instead.

In addition to improving the section-based approach of user preferences, the existing article links could be utilised further. In the current approach they are used only to filter out too similar articles. By building networks based on them, the similarities of articles could be studied deeper. This new similarity information could be used as an additional part of user preferences.

Lastly, one option worth mentioning is to increase the acceptance of the suggestions by adding transparency. It would be relatively easy to show the users their preferences or even give the possibility to change the parameters to some extent. Nevertheless, there are many other means to affect the recommender system's overall success, but each new feature also brings new challenges.

Chapter 7

Summary

The purpose of this thesis was to build a recommender system for news delivery. The system was expected to adapt fast because of the rapidly changing nature of items and users in the news domain. If successful, the suggestions provided by the system were intended to replace the main content on the front page of the news site.

A score for each article was calculated in order to select and order personalised suggestions for each user. For a basis of scores and to separate articles from each other, an article importance score was calculated for each article based on its behaviour on the manual front page. A straightforward approach for user preferences and additional user and item data was combined with the article importance to calculate the final suggestion scores. Even though there are many different kind of successful approaches presented in the literature, a quite unconventional approach was selected because of restrictions set by the media company.

The success of the recommender system was measured as increase in user activity according to five different quantitative performance metrics. Additional goals were to maintain the general feel of the site and not create filter bubbles for the users. The system provides diverse recommendations for users fast. However, the results from the performance metrics were inconclusive:

some metrics showed positive, others negative results. As a conclusion, there is no clear increase in user activity, even though the system does provide recommendations and other goals were achieved.

Multiple ways of improving the building process were identified. The most limiting aspect was the cost of using time for analysis and testing of different approaches and parameters, particularly of offline evaluation and controlled online experiments. For example, it would have been beneficial to compare different recommender system methods or techniques in the beginning, but it was not possible given the constraints.

Because no parameter combination to provide positive results with all or even most metrics was found, the overall evaluation of the system's success remains inconclusive. It could be further assessed by running more controlled online experiments. Additionally, many different means, such as the use of relative user preferences, were proposed for future development of the recommender system in order to achieve better results.

References

G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

X. Amatriain and J. M. Pujol. Data mining methods for recommender systems. In: F. Ricci et al. (eds.), *Recommender Systems Handbook*, pages 227–264. Springer, 2015.

S. S. Anand and B. Mobasher. Intelligent techniques for web personalization. *Proceedings of the 2003 International Conference on Intelligent Techniques for Web Personalization*, pages 1–36. Springer-Verlag, 2003.

V. R. Basili. The role of experimentation in software engineering: Past, current, and future. *Proceedings of the 18th International Conference on Software Engineering*, pages 442–449. IEEE Computer Society, 1996.

G. Beliakov, T. Calvo, and S. James. Aggregation of preferences in recommender systems. In: F. Ricci et al. (eds.), *Recommender Systems Handbook*, pages 705–734. Springer, 2011.

J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

R. Burke. Hybrid web recommender systems. In: P. Brusilovski et al. (eds.), *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 377–408. Springer, 2007.

R. Burke and M. Ramezani. Matching recommendation technologies and domains. In: F. Ricci et al. (eds.), *Recommender Systems Handbook*, pages 367–386. Springer, 2011.

M. Ciobanu. Nzz is developing an app that gives readers personalised news without creating a filter bubble. Webpage, March 3, 2017. `https://www.journalism.co.uk/news/nzz-is-developing-an-app-that-gives-readers-personalised-news-without-creating-a-filter-bubble/s2/a700550/`. Accessed 6.3.2017.

M. de Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro. Semantics-aware content-based recommender systems. In: F. Ricci et al. (eds.), *Recommender Systems Handbook*, pages 119–160. Springer, 2015.

European Comission. Protection of personal data. Webpage, 2016. `http://ec.europa.eu/justice/data-protection/`. Accessed April 28, 2017.

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

A. Gunawardana and G. Shani. Evaluating recommender systems. In: F. Ricci et al. (eds.), *Recommender Systems Handbook*, pages 265–308. Springer, 2015.

I. Himelboim and S. McCreery. New technology, old practices: Examining news websites from a professional perspective. *Convergence*, 18(4):427–444, 2012.

T. Joachims. Optimizing search engines using clickthrough data. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142. ACM, 2002.

R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: Survey and practical guide. *Data Mining and Knowledge Discovery*, 18(1):140–181, 2009.

Y. Koren and R. Bell. Advances in collaborative filtering. In: F. Ricci et al. (eds.), *Recommender Systems Handbook*, pages 77–118. Springer, 2015.

X. N. Lam, T. Vu, T. D. Le, and A. D. Duong. Addressing cold-start problem in recommendation systems. *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication*, pages 208–211. ACM, 2008.

L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 661–670. ACM, 2010.

L. Li, D. Wang, T. Li, D. Knox, and B. Padmanabhan. Scene: A scalable two-stage personalized news recommendation system. *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 125–134. ACM, 2011.

L. Li, L. Zheng, F. Yang, and T. Li. Modeling and broadening temporal user interest in personalized news recommendation. *Expert Systems with Applications*, 47(7):3168–3177, 2014.

G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

J. Liu, P. Dolan, and E. R. Pedersen. Personalized news recommendation based on click behavior. *Proceedings of the 15th International Conference on Intelligent User Interfaces*, pages 31–40. ACM, 2010.

V. Maccatrozzo. Burst the filter bubble: Using semantic web to enable serendipity. *The Semantic Web–ISWC 2012*, pages 391–398, 2012.

S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, pages 1097–1101. ACM, 2006.

E. Mitchelstein and P. J. Boczkowski. Between tradition and change: A review of recent research on online news production. *Journalism*, 10(5): 562–586, 2009.

T. T. Nguyen, P.-M. Hui, F. M. Harper, L. Terveen, and J. A. Konstan. Exploring the filter bubble: The effect of using recommender systems on content diversity. *Proceedings of the 23rd International Conference on World Wide Web*, pages 677–686. ACM, 2014.

D. H. Park, H. K. Kim, I. Y. Choi, and J. K. Kim. A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11):10059–10072, 2012.

M. J. Pazzani and D. Billsus. Content-based recommendation systems. In: P. Brusilovski et al. (eds.), *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 325–341. Springer, 2007.

A. Popescul, D. M. Pennock, and S. Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 437–444. Morgan Kaufmann Publishers Inc., 2001.

P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

F. Ricci, L. Rokach, and B. Shapira. Recommender systems: Introduction and challenges. In: F. Ricci et al. (eds.), *Recommender Systems Handbook*, pages 1–36. Springer, 2015.

F. Rodrigues. Meet the swedish newspaper editor who put an algorithm in charge of his homepage. Webpage, March 21,

2017. http://www.storybench.org/meet-swedish-newspaper-editor-put-algorithm-charge-homepage/. Accessed 25.3.2017.

M. Rodriguez, C. Posse, and E. Zhang. Multiple objective optimization in recommender systems. *Proceedings of the Sixth ACM Conference on Recommender Systems*, pages 11–18. ACM, 2012.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295. ACM, 2001.

J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 158–166. ACM, 1999.

A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260. ACM, 2002.

A. Spangher. Building the next new york times recommendation engine. Webpage, August 11, 2015. https://open.blogs.nytimes.com/2015/08/11/building-the-next-new-york-times-recommendation-engine/. Accessed 6.3.2017.

N. Thurman and S. Schifferes. The future of personalization at news websites: Lessons from a longitudinal study. *Journalism Studies*, 13(5-6):775–790, 2012.

C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Springer Science & Business Media, 2012.